

University of Southern California  
Viterbi School of Engineering  
Introduction to Digital Image Processing  
2018 Spring EE569

Homework #1  
Hang Dong  
Student ID: 8151308667  
[donghang@usc.edu](mailto:donghang@usc.edu)  
January 29, 2018

## Problem 1: Basic Image Manipulation (Used C++)

### (a) Color Space Transformation

#### 1. Abstract and Motivation

We use color model (mathematical model) to describe true world colors by numbers value. Generally human use RGB and CMYK to describe a color space. Specifically, RGB uses three-primary color mixing, CMYK uses reverse color mixing, which used in the printing process often. Different color spaces are adopted for different aims. This problem I tested some methods to change the color space or color distribution of different images.

#### 2. Approach and Procedures

##### (1) Color-to-Grayscale Conversion

There three general strategies to get the grayscale image of a RGB color image as follow:

**a.** Lightness method: Average the most prominent and least prominent colors.

$$\text{grayscale value} = (\max(R, G, B) + \min(R, G, B)) / 2$$

**b.** Average method: Average the values.

$$\text{grayscale value} = (R + G + B) / 3$$

**c.** Luminosity method: Form a weighted average to satisfy human visual effect.

$$\text{grayscale value} = 0.21 R + 0.72 G + 0.07 B$$

##### (2) CMYK Color Space

The Cyan-Magenta-Yellow-(Black) (CMY(K)) color space is another color space that frequently used printing due to the material effect, it is defined by the equation as follows in order to obtain CMY space color value by changing the value of RGB color space.

$$\begin{cases} C = 1 - R \\ M = 1 - G \\ Y = 1 - B \end{cases}$$

The value of R,G,B in the original image has been normalized between 0 to 1. If the image haven't used normalization method we should use 255 to exchange the 1 in each formula.

#### 3. Experimental Results

##### (1) Color-to-Grayscale Conversion



Image.1 Original



Image.2 Lightness method



Image.3 Average method



Image.4 Luminosity method

## (2) CMY(K) Color Space

Result of Bear.raw



Image.5 Bear Original



Image.6 Bear CMK

Result of Dance.raw



Image.7 Original



Image.8 CMK

#### 4.Discussion

##### (1) Color-to-Grayscale Conversion

Image.2 to Image 4 show different grayscale results by different methods. It is obvious that Lightness method is the easiest way but the result is not good enough. Image.4 is better for human eye views and has best result compared other two method.

##### (2) CMY(K) Color Space

I used two different Images finish this experiment and as shown in Image.6 and Image.8 the CMYK color space result is linear changed from the original RGB color space.

##### (b) Image Resizing via Bilinear Interpolation

###### 1.Abstract and Motivation

The main idea of image resizing is changing the size of a picture. One of popular resizing method is bilinear interpolation. The way performing linear interpolation as follow. Firstly, in horizontal direction do linear interpolation and then do it again in the vertical direction.

###### 2.Approach and Procedures



Use the bilinear interpolation to re-size the input color image airplane of size 512x512 to an output image of size 650x650. The re-size method steps :

#1 step, I need to change the coordinate of each pixel. According to the size of input and output image is  $n * n$  and  $m * m$  so the resizing ratio is  $r = \frac{n*n}{m*m}$ .

#2 step, for a pixel  $A(x,y)$  in the new image, we can use the resizing factor to reject this pixel to the pixel in original image  $B(x/t,y/t)$ . The specific calculate formula as follow:

$$f(i+u,j+v) = (1-u) * (1-v) * f(i,j) + (1-u) * v * f(i,j+1) + u * (1-v) * f(i+1,j) + u * v * f(i+1,j+1)$$

### 3.Experimental Results

	
Image.9 Airplane 512*512	Image.10 Airplane_resize 650*650

### 4.Discussion

By comparing the Image.9 and Image.10 it is obviously the original Image was resized to larger scale and the Bilinear Interpolation can protect the Image detail carefully without obvious distortion.

## Problem 2: Histogram Equalization

### (a) Histogram Equalization

#### 1.Abstract and Motivation

Histogram equalization is a vital way for adjusting image intensity to enhance contrast, which is key factor to make the objects in the image distinguishable. The main idea of Histogram Equalization is re-distribute all pixels to an uniform distribution color space.

#### 2.Approach and Procedures

Two methods to implement histogram equalization.

Method A: Transfer-function-based method

Method B: cumulative-probability-based method. Using Bucket algorithm here.

(1)Transfer-function :

One choice of transfer function is the cumulative-distribution function.

#1 step, I statistically calculate the histogram distribution of each gray level in the original image.

#2 step, use the histogram building up the cumulative histogram of each gray scale.

#3 step, the grayscale value mapping relationship between the original image and the new image is obtained and is calculated based on the cumulative histogram distribution.

(2) CPF:

The algorithm of cumulative-probability-based histogram equalization is bucket filling algorithm. Through this progress, the main idea is to make the number of pixels in every grey-scale value equal to  $\text{pixels num}/256$ . By the way, which pixels should change their values is selected randomly.

### 3.Experimental Results

Results of Desk.raw

The original image and histograms of the red, green and blue channels of the original image (Desk.raw) as follow:



Image.11(a) Original

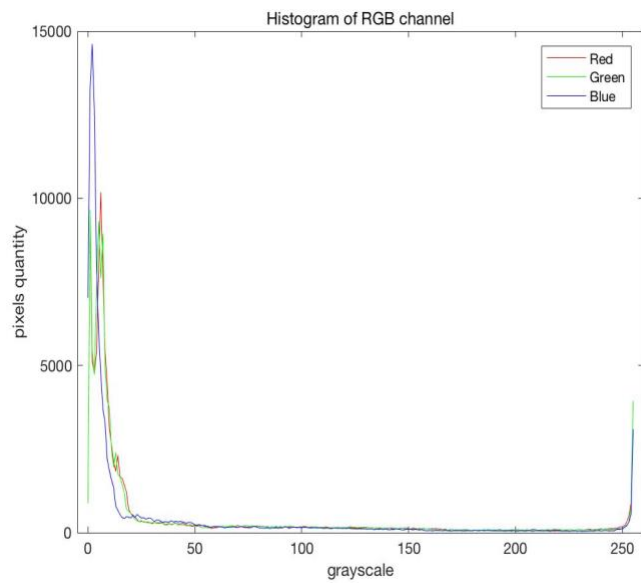


Image.11(b) Histogram of Desk.raw



Image.12(a) transfer-function-based method

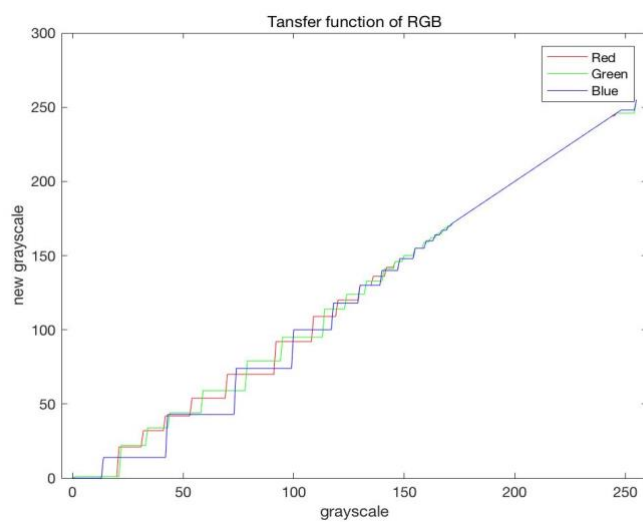


Image.12(b) transfer-function for each channel





Image.13(a) cumulative-probability-based method

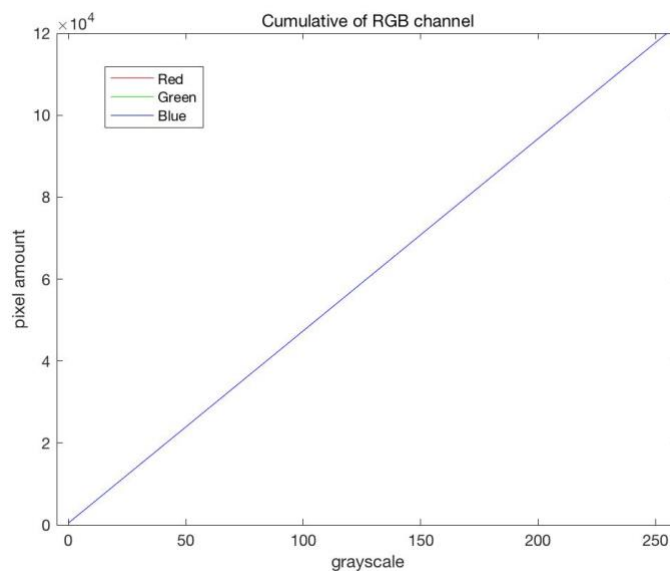


Image.13(b) cumulative histogram for each channel

#### 4.Discussion

By comparing Image.11(a), Image.12(a) and Image.13(a), It is obvious that both Transfer Function and CDF are useful to enhance contrast of image. As shown in Image.12(a) and Image.13(a) the two method have similar results. To enhance the performance of the algorithm, we can implement separate the image into a few of patches and implement method A/method B separately in each patch.

(b) Image filtering-oil effect:

##### 1.Abstract and Motivation

By using color quantization I can get an image similar to the original one but with



few color space. But the oil effect image is different from the original image and has blurring effect.

## 2.Approach and Procedures

The oil effect can be implemented by two steps.

#1 step: Color quantization. Color quantization is a technology that combine the less important similar colors in the original image into one color by reducing the color value and the visual effect of the new image are similar as the original one. Use M values in each channel to represent the original 256 gray scales.

#2 step: For each pixel of the output image of step 1, select the most frequent color in its NxN neighborhood (N is an odd number, usually ranging from 3 to 11), as the representative color for this pixel in another output image

## 3.Experimental Results

Results of Trojans.raw(720\*480)

Parameters: N=3,5,7,9;M=4(64),8(512);



Image.14(a) Trojans Original



Image.14(b) Trojans 64values M=4

Follow 4 pictures are the oil effect.



Image.15 Trojans\_64, N=3,M=4

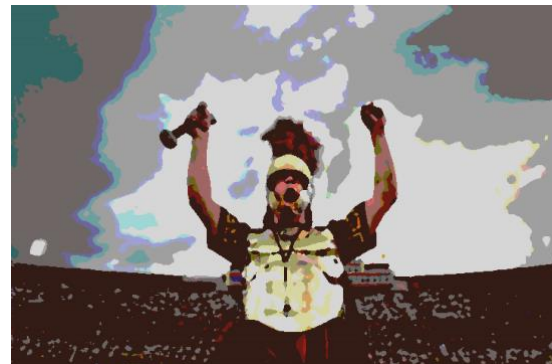


Image.16 Trojans\_64, N=5, M=4



Image.17 Trojans\_64, N=7, M=4



Image.18 Trojans\_64, N=9, M=4



Image.19(a) Trojans 512values, M=8



Image.19(b) Trojans\_512, N=3, M=8

Results of Star\_Wars.raw(720\*480)

Parameters: N=3,5,7,9;M=4(64),8(512);



Image.20(a) Star\_Wars Original



Image.20(b) Star\_Wars 64values, M=4

Follow 4 pictures are the oil effect.

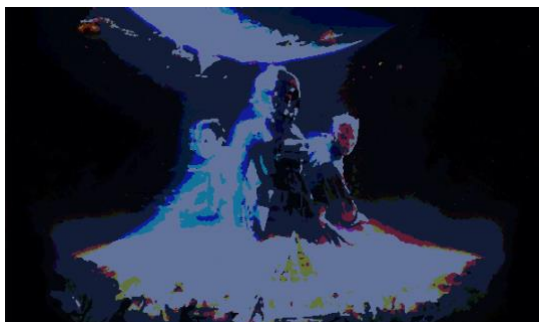


Image.21 Star\_Wars \_64, N=3,M=4



Image.22 Star\_Wars \_64, N=5,M=4

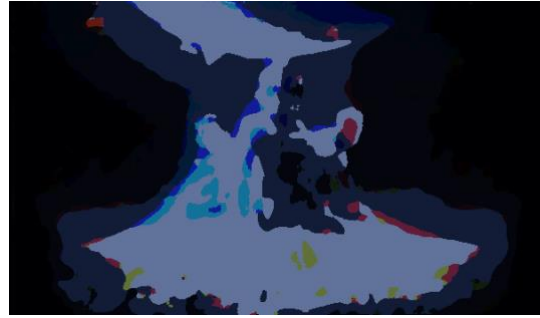


Image.23 Star\_Wars \_64, N=7,M=4



Image.24 Star\_Wars \_64, N=9,M=4

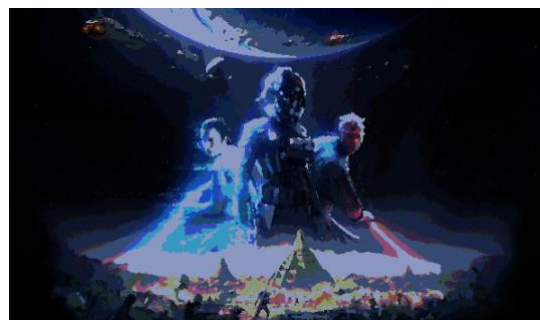


Image.25 Star\_Wars 512values ,M=8      Image.25 Star\_Wars \_512, N=3,M=8

#### 4. Discussion

By comparing the oil effect results of different N(neighborhood size), it is obvious that with the window size increase the image become more and more blurred. What is more, with more values in every channel the oil effect will get bad and the image become clear than the result of image with few values in every channel.

#### (c) Image filtering-film effect:

##### 1. Abstract and Motivation

By comparing the Original.raw and Film.raw, I designed an algorithm to build up the film effect by three steps. This film effect image used CMYK color space and made horizontal mirror of the original image. What is more, by comparing the two images' channel histograms I known the algorithm also do linear mapping to the grayscale in each channel.

##### 2. Approach and Procedures

To begin with, I get the histogram for the example images as follow:



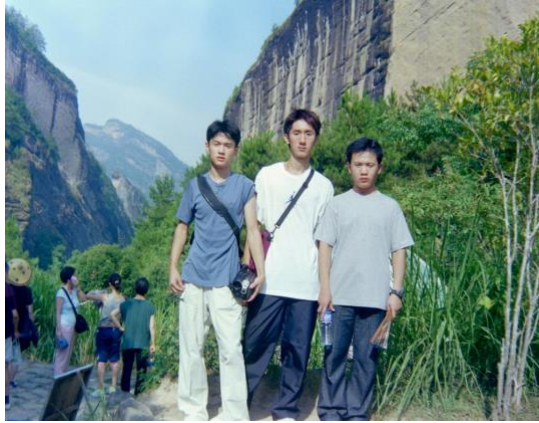


Image.26(a)example\_original



Image.26(b)example\_film

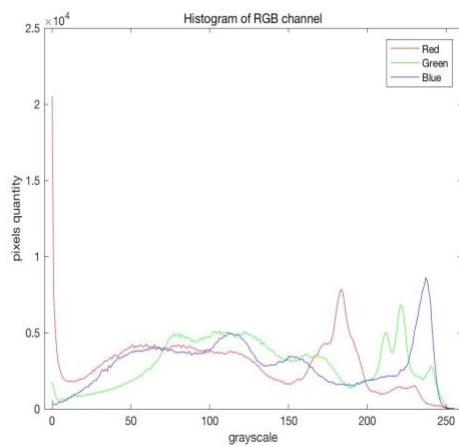


Image.26(c)Histogram\_original

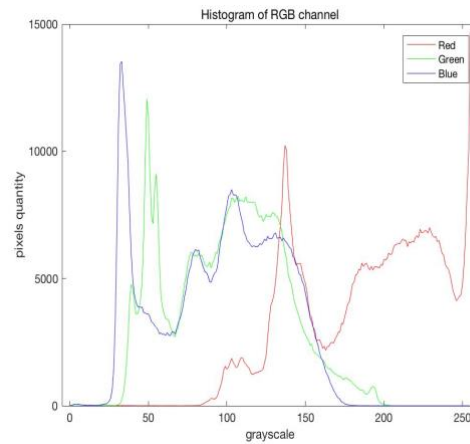


Image.26(d)Histogram\_film

My observation of the film effect is that it used color inversion and mirror. Then by comparing the histogram of grayscales of each channel, the film effect also has grayscale compression.

The algorithm I designed has three steps.

#1 step: Do horizontal mirror transfer and get the new image.

#2 step: Using CMYK method to change the image color space from RGB to CMYK.

#3 step: compression grayscales and linear transform;

R channel:

$$g(i,j)=(3.2/5)*f(i,j)+256*(1.8/5);$$

G channel:

$$g(i,j)=(3/5)*f(i,j)+256*(2/20);$$

B channel:

$$g(i,j)=(3/5)*f(i,j)+256*(2/25);$$

### 3.Experimental Results



Image.27(a) Girl Original



Image.27(b) Girl Film\_effect

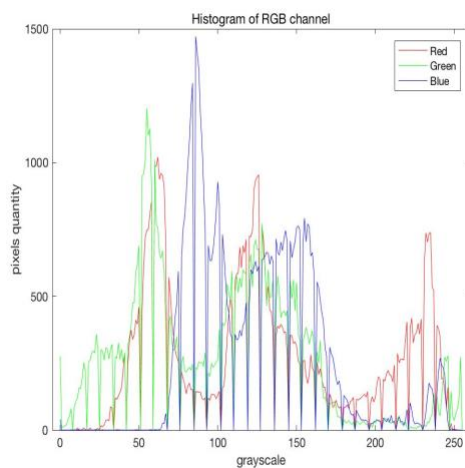


Image.27(c) Histogram\_original

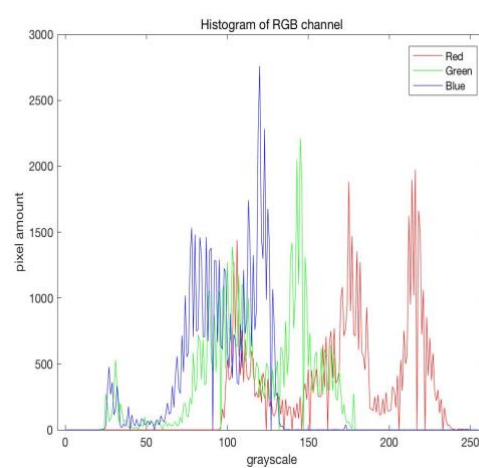


Image.27(d) Histogram\_film

#### 4.Discussion

By comparing Image.26(a) and Image.26(b) I can find that the film effect used the CMYK color space of original image and truly is horizontally mirror of original. According to my result Image.27(b) and its histogram Image.27(d) I am sure my algorithm is similar to the example.

### Problem 3: Noise Removal

#### 1. Abstract and Motivation

Noise prevents people from understanding the information of an image and it's the main reason of image interface. Noise can be separated into some different types, such as salt-and-pepper, Gaussian. To remove noise from the mixed image I need identify different types of noise in every channel and implement different

methods or filters.

## 2. Approach and Procedures

Usually methods to remove noise include median filter, mean filter, low-pass filter, principal component analysis(PCA), Block matching and 3-D (BM3D) transform filter. By comparing the output images of different algorithms and original image, using PSNR to specifically represent the results performance. What is more, a better noise removal algorithm to get a higher PSNR value.

The formula of PSNR calculation:

$$\text{PSNR(dB)} = 10 \log_{10} \left( \frac{\text{MAX}^2}{\text{MSE}} \right)$$

$$\text{MSE} = \frac{1}{NM} \sum_{i=1}^N \sum_{j=1}^M (Y(i,j) - X(i,j))^2$$

X: Original Noise – free Image of size N \* M

Y: Filtered Image of size N \* M

MAX: Maximum possible pixel intensity = 255

Gaussian filter to remove Gaussian noise

The formula of Gaussian filter:

$$\text{value}(x,y) = e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

Different Gaussian mask as follow:

$\sigma = 0.8$ , mask\_size = 3.

$$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

$\sigma = 0.4$ , mask\_size = 3.

$$\frac{1}{614} \begin{bmatrix} 1 & 23 & 1 \\ 23 & 518 & 23 \\ 1 & 23 & 1 \end{bmatrix}$$

Mean filter to remove Gaussian noise(not used in the report)

#1 step, choose a N\*N mask/window and calculate the mean value of the covered pixels' grayscales.

#2 step, use the mean value of the window as the central pixel's new value and write it into the new image space.

Median filter to remove salt-and-pepper noise(outliers)

#1 step, choose a  $N \times N$  mask/window and sort the grayscales of the covered pixels.

#2 step, use the median value of the window as the central pixel's new value and write it into the new image space.

### 3. Experimental Results

(a) Mix noise in color image

Answer of questions

**Part A:** Identify noise types in the noisy Lena image, and answer the following questions.

(1) Do all channels have the same noise type?



Noised Image

R channel

G channel

B channel

No. By viewing the grayscale picture of each channel, it is clear that every channel has salt-and-pepper noise. But Gaussian noise needs more discussion.

By testing the PSNR of the output of median-filter and Gaussian filter, I can get the result as follows (Using matrix size  $N=5$ ):

Channel	Red	Green	Blue
Median-filter output(PSNR)	25.1587	26.637	25.3403
Gaussian-filter output(PSNR)	25.4033	27.0586	25.9218

It is obvious that the PSNR improved a lot in Green and Blue channels. Gaussian noise is mainly distributed in Green and Blue channels. But by using Gaussian-filter, all channels have better PSNR, so I will implement this filter on all channels after

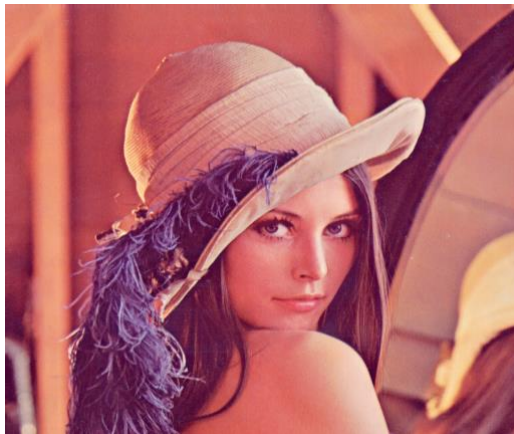


median-filter.

- (2) Should you perform filtering on individual channels separately for both noise types?

Yes, I will do de-noising filters in every channel separately.

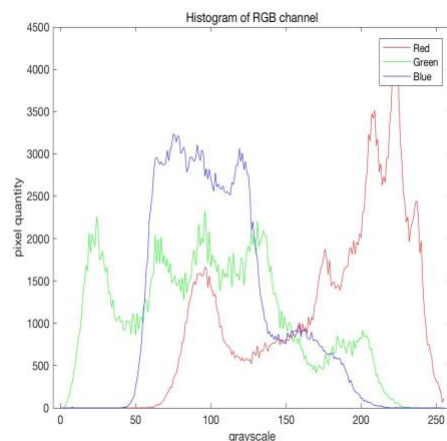
- (3) What filters would you like use to remove mixed noise?



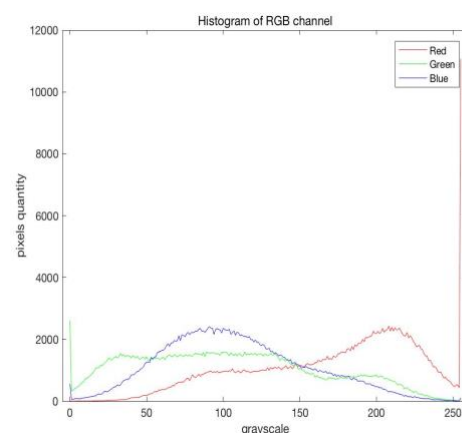
Original Image



Mixed Noisy Image



Histogram of Original Image



Histogram of Noisy mixed Image

According to the histograms of original image and mixed noisy image, It is obvious that the later one has impulsive noise at grayscale=0, 255. That is the feature of salt-and-pepper noise. What is more, there are more uniform than the original image histogram, which is the feature of Gaussian noise. I prefer use median filter to remove salt-and-pepper noise and use Gaussian filter to remove Gaussian noise.

- (4) Can you cascade these filters in any order? Justify your answer.

My steps in order: #1 remove salt-and-pepper noise; #2 remove Gaussian noise. The reason of this order is that salt-and-pepper noise has lots of outliers pixels,

which will deeply influence the performance of Gaussian filter or mean filter. So I need remove salt-and-pepper noise firstly.

(5) Discuss the effect of different filter window sizes.

The window(mask) size of filter also influence the performance of de-noising performance. I tested different window size and then calculate the PSNR of each method.

This sheet is the median-filter output PSNR with different matrix size N.

Channel	median-filter output Image	Average	Red	Green	Blue
N=3		24.7632	23.8739	26.5418	23.8739
N=5		25.712	25.1587	26.6370	25.3403
N=7		25.4349	25.1322	25.6704	25.2023

It is obvious that when windows size is 5\*5 will get the best result.

**Part B:** Get the best results in removing mixed noise.

(1) Describe your method and show its results.

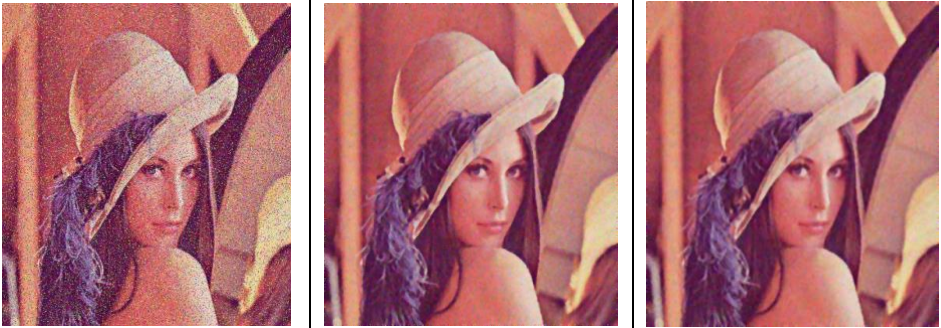
#1 step, choose a mask size and replace the mask center value by the median

value of each pixel value covered by the mask. Slipping the mask continually and searched all the pixels of the image. After this change, I can remove most of salt-and-pepper noise.

#2 step, designed a Gaussian filter based on sigma. I designed the Gaussian filter based on 3\*3 mask matrix. (The Gaussian filter edge will get sharp when sigma decreasing). Then implement it to the image and get the result.

I choose sigma=0.8 as the parameter to build up the Gaussian filter. The follow sheet is the result after Gaussian filtered based on the output image of median-

Gaussian-filter parameter:  $\frac{1}{614} \begin{bmatrix} 1 & 23 & 1 \\ 23 & 518 & 23 \\ 1 & 23 & 1 \end{bmatrix}$

Image			
Image name	Lena_mixed	Median-filter output (Mask=5*5)	Gaussian-filter output(sigma=0.8)
PSNR(avg)	16.8658	25.7120	26.1279
PSNR(R)	17.0757	25.1587	25.4033
PSNR(G)	16.5321	26.6370	27.0586
PSNR(B)	16.9897	25.3403	26.0531

According to the PSNR of the Gaussian-filter output image, the mask size of median-filter N=5 get the best result and all channels have better result compared to the result of just used median-filter.

(2) Discuss its shortcomings.

This method can not remove most of the noise, in other word, its efficiency is limited. It based on the experienced sigma to remove the Gaussian noise. What is more Gaussian-filter will result blurring edge.

(3) Give some suggestions to improve its performance.

We can implement patch to cut the image into lots of small images and calculate the sigma of different patches. Then implement different sigma to build up Gaussian filters and handle the specific patch. What is more, we also can use Bilateral-filter to remove Gaussian noise, this way will not result in blurring edge.

(b) Principal component analysis (PCA) [1][4]

(1). Explain why PCA can be used as a filtering approach for noisy data. What do the components mean and how to choose the number of components in image de-noising.

The main idea of Principal Component Analysis is using linear transformation to reduce the dimension of the original high dimensional data to low dimensional data. By PCA we will lost some not important information of the original data, which usually the noise in the image. After that we use the reduced covariance matrix rebuild the image by reverse transform. The way to choose number of components are variable but follow the principle that all components should over 80%. The usually ways are Bayesian model selection.

(2). Describe and implement the patch-based local PCA (PLPCA) [1]

PLPCA mainly has three steps.




#1 step: Choose some local patches and use Linear transform to get the covariance matrix of the original image.




#2 step: Using the covariance matrix to change from high dimensions to low dimensions. By this way we can maintain the most useful information in the image.


#3 step: Using reverse transform to rebuild the image with few noise.

(3). Apply the PLPCA to House\_noisy.raw (Gaussian noise, sigma= 25) Optimize different parameters, e.g. patch size, the number of components, etc., and discuss their effect on the denoising process. State the best PSNR and your optimized parameters in your report.

Firstly, I tested different patch size and use the factor\_threshold=2.75. The results as follow:


Image name	Image	Patch Size	PSNR(db)
House_noisy		null	20.281
House_plpca_4	<p>PLPCA: PSNR 27.10</p> 	4	27.10
House_plpca_8	<p>PLPCA: PSNR 30.40</p> 	8	30.40

House_plpca_12	<p>PLPCA: PSNR 31.01</p> 	12	31.01
House_plpca_16	<p>PLPCA: PSNR 31.19</p> 	16	31.19
House_plpca_20	<p>PLPCA: PSNR 31.18</p> 	20	31.18




House_plpca_30	<div>PLPCA: PSNR 30.71</div> 	30	30.71
----------------	---	----	-------

As a view of the experiment results, there will be an exact patch size that can get best PSNR result and this image will get the best PSNR=31.19 when patch size=16. Usually based on the formula to choose component:  $(\text{Signal variance}/\text{noise variance}) = 2.5 \sim 3$ . The patch size based on the noise level.

Secondly, Aim to test the influence of different number of components, I change the threshold to analyze it. With threshold increasing there will be few components. I use patch size=16 and test different factor\_threhold as follow:

Image name	Image	factor_threhold	PSNR(db)
House_noisy		null	20.281






House_plpca_16_ 2	<p>PLPCA: PSNR 28.43</p> 	2.00	28.43
house_plpca_16	<p>PLPCA: PSNR 31.19</p> 	2.75	31.19
House_plpca_16_ 3	<p>PLPCA: PSNR 31.27</p> 	3.00	31.27

House_plpca_16_ 3.5		3.50	30.95
------------------------	--	------	-------

According to the experiment result I can observe that when the factor\_thrhold change to 3 can get the best result PSNR=31.27. Others no matter larger or smaller the PSNR performance will decrease.

(4). Apply the approach in part (a) to House\_noisy.raw and compare the performance of the PCA approach with those of filters used in part (a). Please explain the advantages of the PCA approach and why.

The result of approach in part (a):

Image			
Image name	House_noisy	Median-filter output (Mask=5*5)	Gaussian-filter output(sigma=0.8)
PSNR	20.2531	27.1933	27.6716

PCA can better keep the edge in the image without blurring edge and get better PSNR result. That due to PCA identify the component of noise by calculate the

data's covariance matrix and analyze it to remove the non-important components.

(c) Block matching and 3-D (BM3D) transform filter [2][3]

(1). Please explain the BM3D algorithm in your own words, and implement the BM3D filter to de-noise the noisy image: House. Discuss the effects of several tunable parameters on the de-noising result.

The BM3D algorithm includes 2 parts, fundamental estimation and final estimation, includes totally six steps.

Fundamental estimation:

#1 step: Grouping:

In order to group some similar blocks into a 3D matrix, we can search  $M \times M$  neighbor area and pick some blocks with minimum differences.

#2 step: Collaborative Filtering:

Do two-dimensional transformation in each 2D block (the directions of image each) and then do transformation in the third dimension of the 3D matrix.

#3 step: Aggregation:

Aggregate all two-dimensional block (which already is the estimation of denoising image) to calculate the pixel value in the original image position.

Final estimation:

#4 step: Grouping:

Generate two 3D matrix, one is got from the original noisy image and one is got from the result of fundamental estimation.

#5 step: Collaborative Filtering:

Using Wiener Filter and the fundamental estimation matrix value we got in step 2 to modify the coefficient in the matrix obtained by original image.

#6 step: Aggregation:

Use the coefficients got in step 5 and the noise parameter in order to calculate the pixel value mapping to the output image.

(2). Why does the author utilize block matching? What is the motivation of step 2 (block matching and Wiener filtering) and compare the de-noising performance with and without step 2.

Block matching can achieve de-noising without blur edge. Blurring edge is the shortcoming of mean-filter. BM3D avoid blurring edge by group matching, which

will pick up some blocks that not neighbored to each other. Using non-local mean to keep the sharpness of the image.

Applying the similar progress twice can get a more effective performance since it applies the result of fundamental estimation, which already has few noise than the original image.

(3). How would you classify BM3D - spatial domain filter, frequency domain filter, or both? Justify your answer.

Both.

Firstly, block grouping is a spatial way due to it searched the spatial domain.

Secondly, it transforms to frequency domain using wavelet transformation and remove the component of noise in frequency domain.

(4). Conduct qualitative performance comparison between the algorithms developed for Problem 3(b) and BM3D.

The experiment result of BM3D as follow:

**Noisy External i, PSNR: 20.281 dB (sigma: 25)**



Image.28(a) The mixed noise image. Sigma=25; PSNR=20.281dB

**Denoised External i, PSNR: 32.676 dB**



Image.28(b) BM3D output image. Sigma=25; PSNR=32.676dB

After BM3D operation, it is obvious that the PSNR improved a lot than the mixed noise image. Compared to the result in Problem 3(b), which has the best PSNR=31.27dB when patch size=16, factor\_threshold=3.00. It is obvious these two algorithms have similar de-noise abilities, but BM3D has better result. That due BM3D do de-noise both in spatial and frequency domains. But in the other hand, BM3D also the most complexity method in all the methods above, which will cost more time to compute the result.

## References

- [1] Deledalle, Charles-Alban, Joseph Salmon, and Arnak S. Dalalyan. "Image denoising with patch based PCA: local versus global." BMVC. Vol. 81. 2011.
- [2] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-d transformdomain collaborative filtering," Image Processing, IEEE Transactions on, vol. 16, no. 8, pp. 2080–2095, 2007.
- [3] [Online]. Available: <http://www.cs.tut.fi/~foi/GCF-BM3D/>
- [4][Online].Available: [http://josephsalmon.eu/code/index\\_codes.php?page=PatchPCA\\_denoising](http://josephsalmon.eu/code/index_codes.php?page=PatchPCA_denoising)