Su Kai

Digital Image Processing
Phone Photo Enhancement

Bachelor's Thesis
Information Technology

May 2013

MIKKELIN AMMATTIKORKEAKOULU
Mikkeli University of Applied Sciences

DESCRIPTION

MIKKELIN AMMATTIKORKEAKOULU Mikkeli University of Applied Sciences	**Date of the bachelor's thesis** 09.05.2013
Author(s) Su Kai	**Degree programme and option** Information Technology

Name of the bachelor's thesis

Digital Image Processing –Phone Photo Enhancement

Abstract

With technology improving, the photos taken by today's phone are better than the phone photos taken years ago. It means that we maybe not satisfied with the photos taken by today's camera phone years later. Another thing, shooting is a good but never be a simple thing. There is a lot of things to consider such as ISO, White balance, Saturation, Metering and so on. Most of time, we fail to get a good image.

The topic of this thesis is analysis this kind of photo in a basic way. Histogram, FFT. Each tool has its own function against these bad images. From histogram diagram, we see the brightness distribution. By FFT we know something about the frequency domain of a image.

As the drawback of JPEG file, the picture itself is not good, some useful information has lost since the image is made. So manually managing these values of the pixels is a good way to try.

Actually, in my thesis, I combine some mathematics when these tools are underway. Such as, make a ellipse filter to fit the frequency spectrogram to reduce more unexpected things.

Subject headings, (keywords)

Histogram, FFT, Color Enhancement.

Pages	Language	URN
39	ENGLISH	

Remarks, notes on appendices

Tutor	Employer of the bachelor's thesis
Reijo Vuohelainen	

CONTENTS

1 INTRODUCTION

1.1 Background

Society now has higher requirements for phone photos and phones. More and more mobile developers consider more high-tech cameras installed in their phone products. It means that the pictures taken by previous mobile phones are not that good, they are in poor quality and they are not clear and in real color. We need do something with these unsatisfied images. Then the process to make this kind of image more real and more colorful is the main task in my thesis.

Well, there are some other problems that may cause the photo taken by phone are not in good quality. One of those is that the camera components of the phone are not good such as the low quality lens will result that the visual resolution of the phone camera is lower than the level declared. Another reason is that not all phone producers using the newest technology at that time.

Most possibility is that when we shoot, we fail to set the right ISO value, white balance, the right shooting mode and so on. Photography is really not that simple. Even though you have a good digital camera, it is still hard to get a good photo while shooting under the sun. Most of the time we fail to get a good picture.

1.2 Aims of the study

The aim of this thesis is to enhance a photo taken under the sun. I want to enhance the color. I need to change the background of the photo.

The way I started my thesis is to compare the blur images and the clear images. A clear image is the sample and the goal. Try to get useful parameters of the clear images and use as measure to enhance the other images. And it is likely to enhance some part of the images.

I will illustrate the principles of color images and how matlab processes them. Both spatial domain transaction and frequency domain transaction will be contained in our thesis. In spatial domain I will focus on Histogram. In frequency domain, smoothing frequency-domain filter need to be figured out.

The image resolution is getting larger. Then to compute a image matrix like 640*480 is really tough task and it cost much time. Some concerns about saving processing time will be included task.

I have two photoes, only one of them is fit to our requirements. The job to change the value of the pixels of the images belonged to each channel is the main direction to process the images. Later, I hope to look for common characters of these blur images which were taken by the same camera phone and within few hours.

Before gain this simple idea, I will collect much pixel information. Because I need to know what these value are when presenting these different colors.

2 BASIC PREPARATION

2. 1 The construction of camera phone

It is easy for us to get a camera phone nowadays. We take them with us every day and sometimes, we use it to take photos to capture what we beloved.

Figure 1. Phone cameras

Figure 1 shows the cameras that are the common built-in digital cameras of our phones for shooting images or movies.

The main difference between a digital camera and a camera phone is that zoom principle of a camera phone is digital zoom. It is not good to get the scene which is a little far away from you. All the job of shooting is controlled by CPU which integrates video processing system and camera driver.

The process that a camera phone gets a photo like this:

Light -> Lens -> image sensor (i.e. the photoreceptor, the light is converted to a digital signal) -> a digital signal processing chip (i.e. the main chip, to optimize the digital signal processing, and transmission and preservation) ->image.

When we shoot in JPEG the camera's internal software (often called "firmware" since it's part of the hardware inside your camera) will take the information off the sensor and quickly process it before saving it. Some color is lost as is some of the resolution and on some cameras there is slightly more noise in a JPEG version than the raw version.

Most camera phones are simpler than separate digital cameras. Lens, sensor and DSP are the three main components of a camera. High quality lens, bigger and stronger image sensor, and high-tech DSP give better quality of images.

2. 2 Photo Formats

The picture we get from our cell phone is in JPG format because most cell phone cameras save images as JPG files by default. Well, JPEG is well known as the ability that it can give 10:1 compression with little perceptible loss in image quality. Then it saves storage and as a result of that JPG images get wide used in network. What's more, JPG Images supports 8-bit grey scale images and 24-bit color images (8 bits each for red, green, and blue). This is very useful in image processing.

So it means that the photo we take by our phones has been compressed by JPFG compression. JPEG is lossy compression and it cannot suit all conditions, such as JPEG may not be as well suited for line drawings and other textual or iconic graphics.

Because our picture have suffered bad lens, low end sensor and DSP, lastly it goes through lossy compression, our picture is not in good quality

2. 3 The introduction of Color image

There are a lot of color models created to record our colorful world. One of them is RGB color model which is based on a Cartesian coordinate system. Images represented in the RGB color model consist of three component images, one for each primary color. RGB also has been developed into several types. According to the number that used to represent each pixel, RGB24 is a standard that uses 8 bits for each color channel just as used in JPG File.

RGB24 color space looks like a cube. Black is at the origin; and white is the corner farthest from the origin.

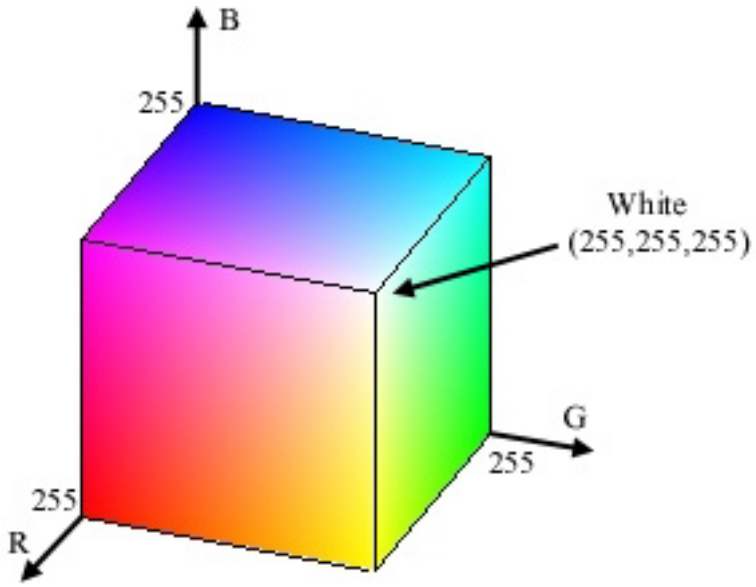


Figure 2. RGB 24-bit color cube

The main purpose of the RGB color model is for the sensing, representation, and display of images in electronic systems, such as televisions and computers, though it has also been used in conventional photography.

A typically process that a color image is acquired by using three filters, sensitive to red, green, and blue, respectively. When we view a color scene with a monochrome camera equipped with one of these filters, the result is that monochrome camera intensity is proportional to the response of that filter.

2. 4 MATLAB With Color Image

Any gray image may be defined as a two-dimensional function. f(x, y), where x and y are spatial coordinates and the value of f at any pair of coordinates (x, y) is called the intensity of the image at that point. Then we have our image matrix:

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & \vdots & & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix} \quad (1)$$

Actually, the matrix start as f(1, 1) not f(0, 0),, and f(M, N) instead of f(M-1, N-1).

An RGB color image is an MxNx3 array of color pixel, where each color pixel is triplet corresponding to the red, green, and blue components of an RGB image at a specific spatial location.

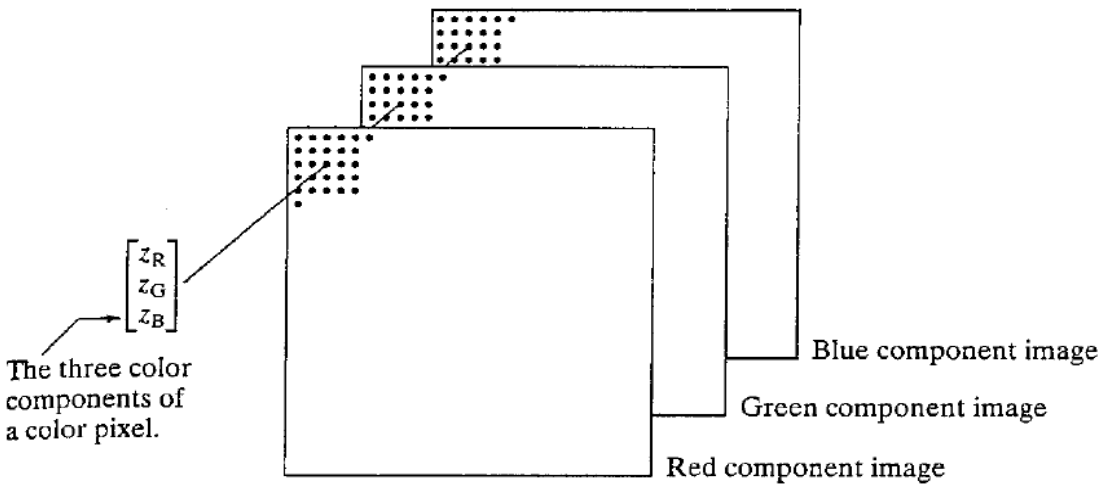


Figure 3. Color image planes as a matlab matrix

What an amazing thing is that these numbers between 0 and 255 in three image matrixes can show different visual effect while they are presenting the same object such a person. In other words, 256*256*256 is 16777216 , we shall have 16777216 kinds of color by changing the numbers in three planes. As a result of the limit of human eyes, we cannot distinguish them from each other, in the condition that the difference between the maximum value and the minimum value can be as large as 20 where the part of an image is visually the same.

In other words, we cannot create these values of a photo of eye but we can copy, move these values. Each pixel has two attribute: position and brightness which we can manage.

2. 5 MATLAB with Indexed Image

An indexed image has two components: a data matrix, X, and a colormap matrix. Matrix map is an m *3 array of class double containing floating-point values in the range [0, 1]. The value of m is the length of the map which is equal to the number of colors it defines. Each row of map specifies the red, green, and blue components of a single color. An indexed image uses ”directing mapping ” of pixel intensity values to colormap values. The color of each pixels is determined by using the corresponding value of integer matrix X as a printer into map. If X is of class double, then all of its components with value 2 point to the second row,

and so on. If X is of class unit8 or uint16, then all components with value 0 point to the first row in map, all components with value 1 point to the second row, and so on. These concepts are illustrated in the following image.

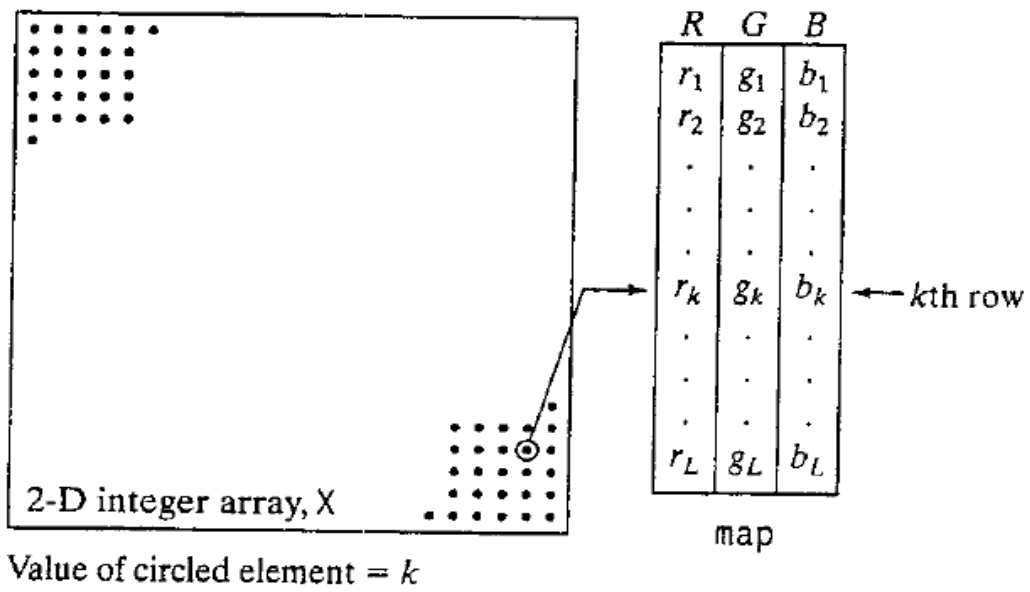


Figure 4. Elements of an indexed image.

Display an indexed image with the statements:

image(X); colormap(map)

A colormap is often stored with an indexed image and is automatically loaded with the image when you use the function imread . However, you are not limited to using the default colormap—use any colormap that you choose.

Here is a list about RGB values of some basic values.

Long name	Short name	RGB values
Black	k	[0 0 0]
Blue	b	[0 0 1]
Green	g	[0 1 0]
Cyan	c	[0 1 1]
Red	r	[1 0 0]
Magenta	m	[1 0 1]
Yellow	y	[1 1 0]
White	w	[1 1 1]

Figure 5. RGB values for basic colors.

Here RGB values for color white is [1 1 1], then it is the same as (255,255,255). As the color was indexed and the amount of colors is limited, we change the colormap to get the value we want. On the other hand, we can copy the values from the other colormap.

3 PRIMARY ANALYSIS OF COLOR IMAGES

3. 1 Histogram analysis

The first step I want to use histogram to help me know more details about the intensity of the original images. Generally, it gives the description of the image brightness level in the two-dimensional charts, the horizontal axis is the brightness, in which dark is on the left and light is on the right, and the vertical axis is number of pixels in the same brightness. As we know, the color image is a 3 dimensional picture. As a result of that, we will get three separate histograms, one each for the R, G and B channels.

This method can be used to increase the many global contrast of the image, especially when the useful data of the image contrast is quite close. By this method, the brightness will be better distributed on the histogram. This can be used to enhance the local contrast without affecting the overall contrast, and by this way we can use histogram equalization several times to make up for the shortcoming that its processing of the data is not selective.

The disadvantage will result in that the increase of the contrast of the background noise and the decrease of the contrast of the useful signal. Well, these will maybe be removed by using noise filters.

Histogram equalization achieves the goal of distributing brightness by effectively expanding top brightness. This method for the background and foreground are too bright or too dark image is very useful. A major advantage of this approach is that it is a fairly intuitive technology and is reversible operation, known equalization function, you can restore the original histogram, and the calculation is not difficult.

When we consider a continuous situation, the basic principle is $p(y)dy=p(x)dx$. As to the process is a little complex, here figure 6 show how the histogram equalization is done.

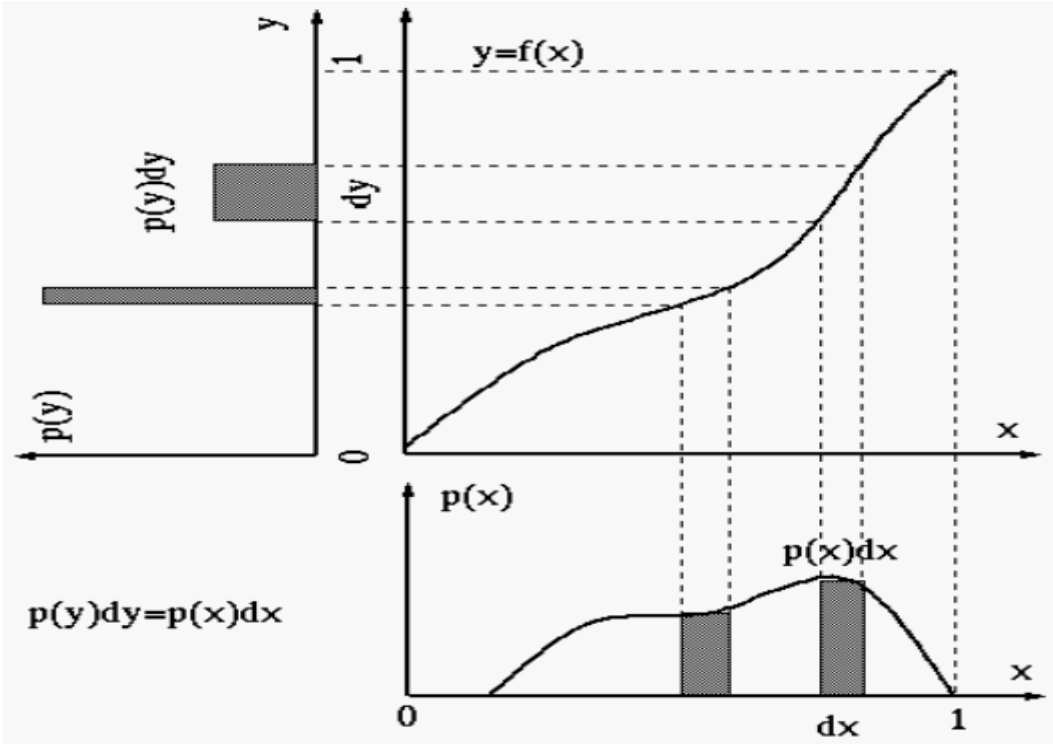


Figure 6. Function Principle

p(y):original image histogram **p(x)**:corrected image histogram **y=f (x)**:transform function

Function f (x) may have different shapes depends on the value of p(x).

When we manage digital values, we deal with discrete values. So there a little different from figure 6, we do with probabilities and summations instead of probability density functions and integrals.

3. 2 Fast Fourier Transforms

After I use special tools, here the transform used to help us to analyse the spectrum is Fast Fourier transform (FFT). FFT is a discrete fast Fourier transform algorithm, which is based on the discrete Fourier transform (DFT) odd, even, virtual and real characteristics of the Discrete Fourier Transform. The most common FFT is Cooley–Turkey algorithm which was introduced by Cooley and Turkey in 1965. An FFT is a way to compute the same result more quickly: typically, computing the DFT of N points in the naive way, take $O(N^2)$ arithmetical operations, while an FFT can compute the same DFT in only O(N log N). When N became bigger, the better effect of time saving can be seen.

The following shows the principle of Fast Fourier Transforms.

The functions Y = fft(x) and y = ifft(X) implement the transform and inverse transform pair given for vectors of length *N* by:

$$X(k) = \sum_{j=1}^{N} x(j)\omega_N^{(j-1)(k-1)}$$
$$x(j) = (1/N)\sum_{k=1}^{N} X(k)\omega_N^{-(j-1)(k-1)} \qquad (2)$$

Where $\omega_N = e^{(-2\pi i)/N}$ is an *N*th root of unity.

As when we deal with image channels we are computing 2-D matrix and we need use fft2. Matlab was an already made function, fft2(X) can be simply computed as fft(fft(X). '). ' That computes the one-dimensional DFT of each column X, then of each row of the result. The execution time for FFT depends on the length of the transform. It is fastest for powers of two. It is almost as fast for lengths that have only small prime factors. It is typically several times slower for lengths that are prime or which have large prime factors.

Mathematically, fft2 comes from the follow way:

$$\begin{aligned} F(u,v) &= \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)e^{-i2\pi(ux/M+vy/N)} \\ &= \frac{1}{M}\sum_{x=0}^{M-1} e^{-i2\pi xu/M}\left(\frac{1}{N}\sum_{y=0}^{N-1} f(x,y)e^{-i2\pi yv/N}\right) \end{aligned} \qquad (3)$$

FFT give me a good approach to analyze the image by frequency. The image frequency is characterized by the indicator of the intensity of the pixel gray value varies. From the physical effects, the Fourier transform of the image converted from the spatial domain to the frequency domain, the inverse transform of the image is converted from the frequency domain to the spatial domain. In other words, the physical meaning of the Fourier transform is a function of the gray level distribution of the image is converted into a frequency distribution of the image function, inverse Fourier transform of the image frequency distribution function transform gray distribution function. FFT transform an image from the spatial domain to the frequency domain, but it doesn’t mean that there is kind of correspondence between the point in an image and the point in frequency spectrum even before use shifting the frequency. On a Fourier spectrum diagram we can see that the varying brightness of the bright spot, in fact, it shows the strength of the image points in the neighbourhood differences, that also is the size of gradient and the size of frequency. In other words, white spot means low frequency, the black spot means high frequency.

FFT is a fast way to compute DFT. The following is something about that how FFT can save time in computing DFT.

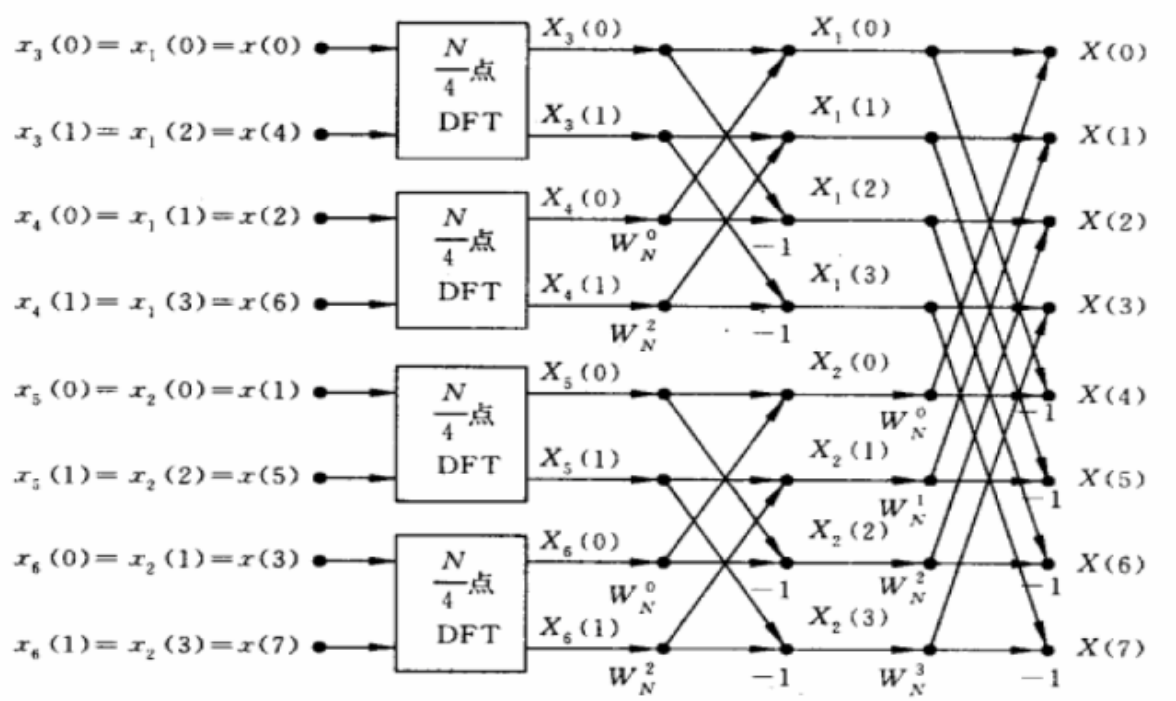


Figure 7. N=8 FFT base-2 butterfly diagram.

Data flow diagram for *N*=8: a decimation-in-time radix-2 FFT breaks a length-*N* DFT into two length-*N*/2 DFTs followed by a combining stage consisting of many size-2 DFTs called "butterfly" operations. Base on the characteristics of W_N that:

Symmetry:

$$\left(W_N^{nk}\right)^* = W_N^{-nk} \tag{4}$$

Periodicity:

$$W_N^{nk} = W_N^{(n+N)k} = W_N^{n(k+N)} \tag{5}$$

Reducibility:

$$W_N^{nk} = W_{mN}^{mnk}, \quad W_N^{nk} = W_{N/m}^{nk/m} \tag{6}$$

We get:

$$W_N^{n(N-k)} = W_N^{(N-n)k} = W_N^{-nk}, \quad W_N^{N/2} = -1, W_N^{(k+N/2)} = -W_N^k \tag{7}$$

So during the computing progress, some values of W_N are the same, there is no need to compute it every time. Then we save time. Compared to DFT which need 8^2=64 multiplication operations and here we only need 8+4+4+4+4=24 multiplication operation in figure 7. In computer, the multiplication operation takes more time than addition and subtraction, so it really reduces processing time.

3.3 Filter

Because the edges of the image, and other sharp jump (such as noise) have contributed much to the high-frequency components of the frequency spectrum. The method that let the image pass a linear system so that a certain range of high-frequency component will get attenuation then the image can be smoothed is called the low-pass filter method.

If f(x, y) is the original image with noise, then F (u, v) is the result of use fft2 on f(x, y). If G (u, v) is the result of F (u, v)H (u, v) , in which H (u, v) stands for the Filter, then the result of using ifft2 on G (u, v) is output image is g(u, v).

Ideal low-pass filter (ILPF)

Definition: All the frequency components within a circle of radius D_0 pass through losslessly, all the frequency components outside of circle will be completely attenuated. The equation 8 and figure 8 show the formula and the shape of BLPF.

$$H(u,v)=\begin{cases}1 & D(u,v)\le D_0\\ 0 & D(u,v)>D_0\end{cases} \qquad (8)$$

$$D(u,v)=\sqrt{u^2+v^2}$$

D_0 is also known as the cutoff frequency.

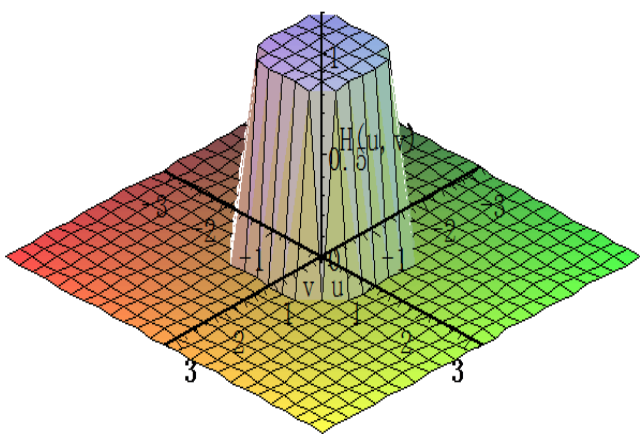


Figure 8. ILPF Model

Because of the sharp step from 0 to 1, the filter will cause the image a little fuzzy, the effect is called the ringing phenomenon.

Here what I want to use is Butterworth low-pass filter. It has advantages:

Firstly, fuzzy phenomenon is greatly reduced. Secondly, there is less ringing phenomenon compared with ILPF because the filter is smooth and continuous. The equation 9 and figure 9 show the formula and the shape of BLPF.

n-order Butterworth Filter

$$H(u,v)=\frac{1}{1+\left(\frac{\sqrt{u^2+v^2}}{D_0}\right)^{2n}} \tag{9}$$

D_0 is also known as the cutoff frequency.

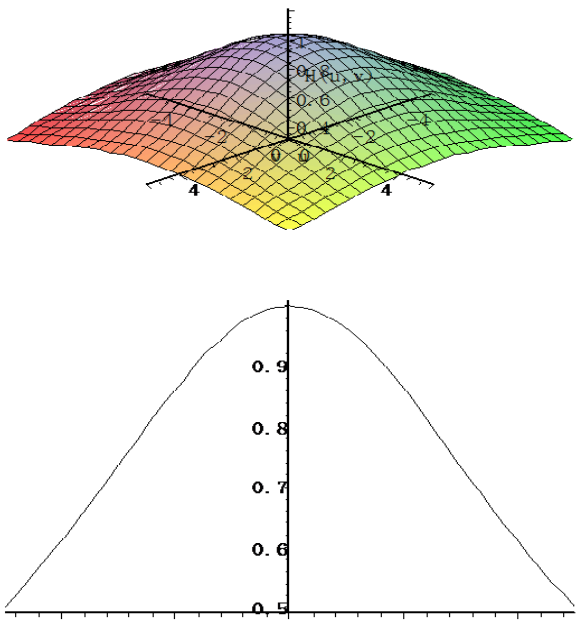


Figure 9. Butterworth Filter model

When it comes to high-pass filter, it works similar way as LP but opposite way, you need to inverse the formula 8. High-pass filter will eliminate high frequency.it will sharpen the image. The equation 9 shows the formula e of IHPF.

$$H(u,v)=\begin{cases}0 & D(u,v)\le D_0\\ 1 & D(u,v)>D_0\end{cases} \tag{10}$$

$$D(u,v)=\sqrt{u^2+v^2}$$

The transform function of Butterworth high-pass filter is equation 11.

$$H(u,v)=\frac{1}{1+\left(\frac{D_0}{\sqrt{u^2+v^2}}\right)^{2n}} \tag{11}$$

I will use and BLPF in this thesis. There are more filters available to use., but I have some idea to change or improve the filter.

4 CODES AND RESULTS

The phone photo which I want to enhancement is shown in figure 10.

Figure10. psue JPG image

Matlab has this kind of function that identifies the attribute of an image.

```
>>imfinfo    psue. jpg        %used to show the information of an image.
ans =
                    Filename: 'F:\My Documents\Mat\2012\psue. jpg'
```

```
        FileModDate: '21-Jan-2013 18:16:22'
            FileSize: 99965
              Format: 'jpg'
               Width: 640
              Height: 480
            BitDepth: 24
           ColorType: 'truecolor'
     FormatSignature: ''
  Number Of Samples: 3
       CodingMethod: 'Huffman'
      CodingProcess: 'Sequential'
```

‘Truecolor’ means RGB24.

CodingMethod is Huffman because Huffman coding is used in JPEG compression progress.

4. 1 Indexed image

Now I change the RGB image to indexed image. And change the number of indexed colors to 32.

```
clc;close all; clear all;
I= imread('psue. jpg');
IB=imcrop(I, [210 60 219 419]);    % get wanted segments
imwrite(IB, 'IB. jpg');         %load the image to current Matlab work filder.
figure, imshow(IB), title('IB');% show the image on the screen.
[IB1, map1] = rgb2ind(IB, 32); %
figure, image(IB1);colormap(map1), title('IB1');
axis off
axis image
```

Here if I use command: imwrite(IB1, 'IB1. jpg'); I will not get a color image because indexed image has two components, of which IB1 I only a data matrix, in other words it is a grey image. I get the image from Matlab figure window by the saving it as JPG file. Then I cut its main part. This method will be used commonly in this thesis.

Here is the comparison shown in figure 11

Figure 11. original image

Figure12. turn to index image.

From the figure 12, we see that there is some visual color spots resulted from the amount of color kinds has reduced. Now indexed image has only 32 kinds of color, then the color can be easily managed.

4.2 Imhist

Imhist function will help us to show the brightness distribution of a gray image. It is easy to get the histogram with MATLAB.

Here are the codes for showing the histogram of Image psue.

```
clc;close all;clear all;
I=imread('psue. jpg');
I1=I(:, :, 1);I2=I(:, :, 2);I3=I(:, :, 3);    % get each color plane.
figure, imhist(I1), title('RED');figure, imhist(I2), title('GREEN');
figure, imhist(I3), title('BLUE');
```

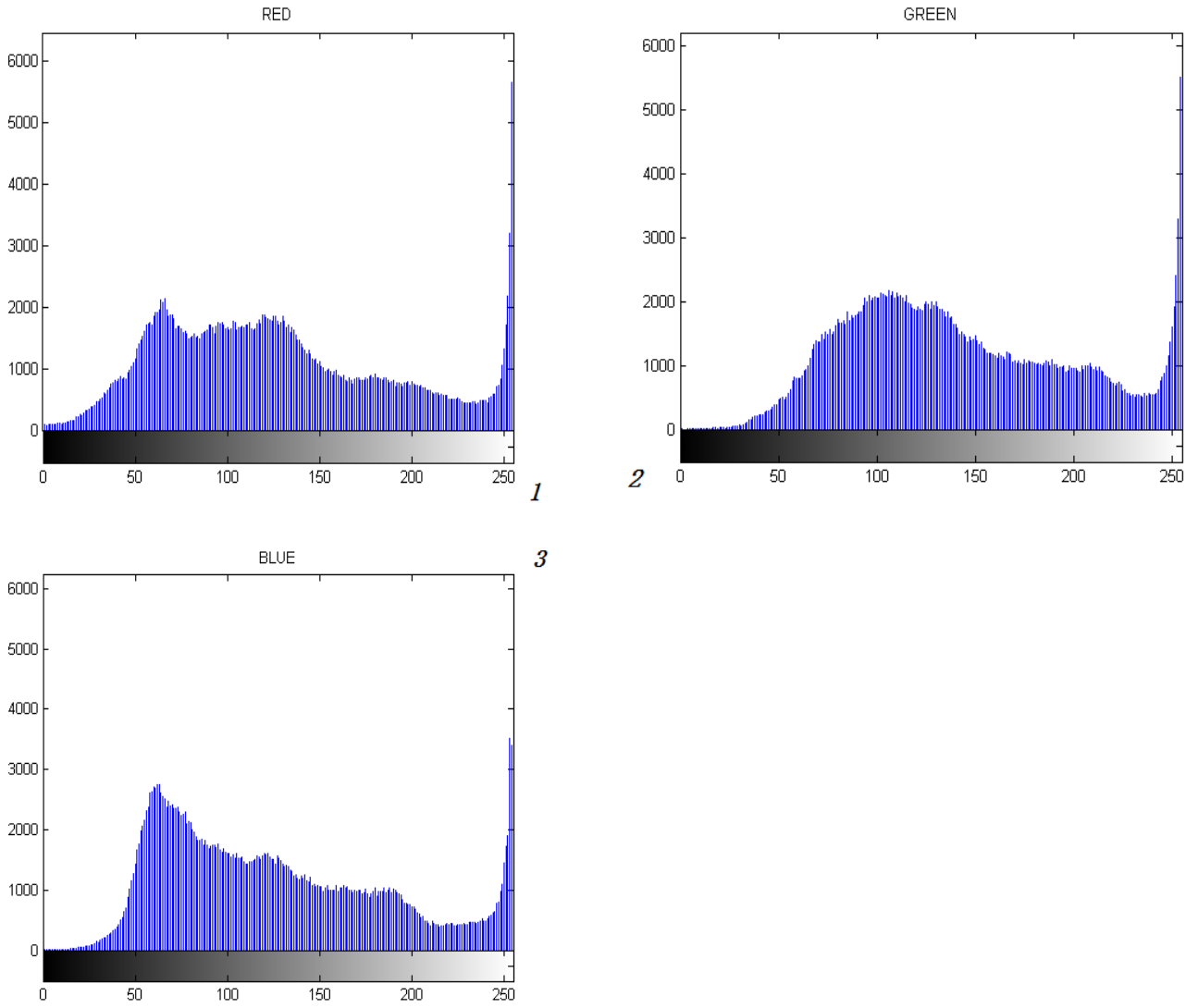


Figure13. Imhist images of three color plane.

Here I need to use command; I1=I(:, :, 1) to get the red plane because imhist function can only deal with 2D image.

From histogram: RED, GREEN, BLUE in figure 13, we can find that the brightness distribution is not even because only only few lower brightness values and more higher brightness values. And the photo is overexposed absolutely. When shooting, the ISO has been set too high.

Table 1. Statistics of the psue image.

	Min amount	Max amount	AVERAGE	ABOVE 240	Percentage
RED	75	3483	141. 39	56184	18.28
GREED	0	3745	154. 23	56186	18.28
BLUE	1	3241	139. 40	52559	17.11

From the table we know that almost one fifth of the image brightness values all above 240, as a result of that it looks like a white veil on the image. So if we can reduce the amount of values over 240, the image will show different.

```
I=imread('psue. jpg');
I1=I(:, :, 1);I2=I(:, :, 2);I3=I(:, :, 3);
h3=imhist(I3);
figure, plot(h3), title('BLUE2');
axis([0 255 0 6000]);
set(gca, 'xtick', [0:16:255])
set(gca, 'ytick', [0:400:6000])
```

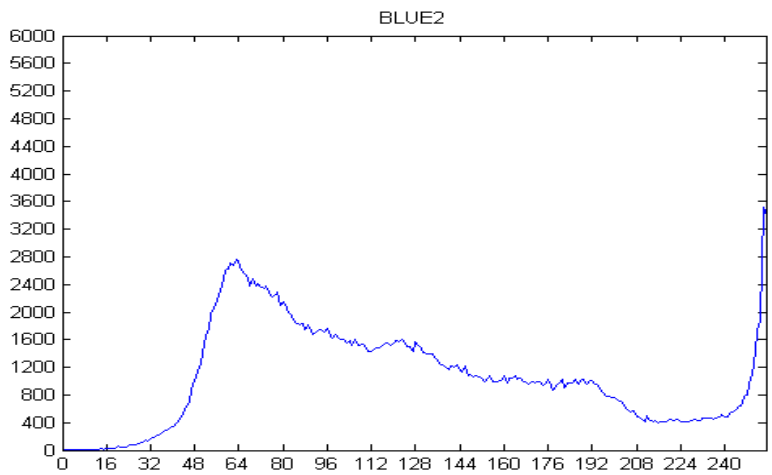


Figure 14. BLUE2

From figure 14, we can see that the brightness value 240 maybe a demarcation point. The pixels at the right side are overexposed. So when we do function imadjust, we may use the value 240 as upper bound to adjust

```
clc;close all;clear all;
I = imread('psue. jpg');
Ir = I(:, :, 1);Ig = I(:, :, 2);Ib = I(:, :, 3);
%40/255=0. 15686;          250/255=0. 98039;        240/255=0. 9412
I2=imadjust(im2double(I), [0.15 0.94], [0.05 0.98], 1.695);
Ir2 = im2uint8(I2(:, :, 1));
Ig2 = im2uint8(I2(:, :, 2));
Ib2 =im2uint8(I2(:, :, 3));% change each color plane back to uint8 format.
figure,
```

```
subplot(311), imhist(Ir2), title('B Image');
subplot(312), imhist(Ig2);     subplot(313), imhist(Ib2);
figure, imshow(I2), title('After');
```

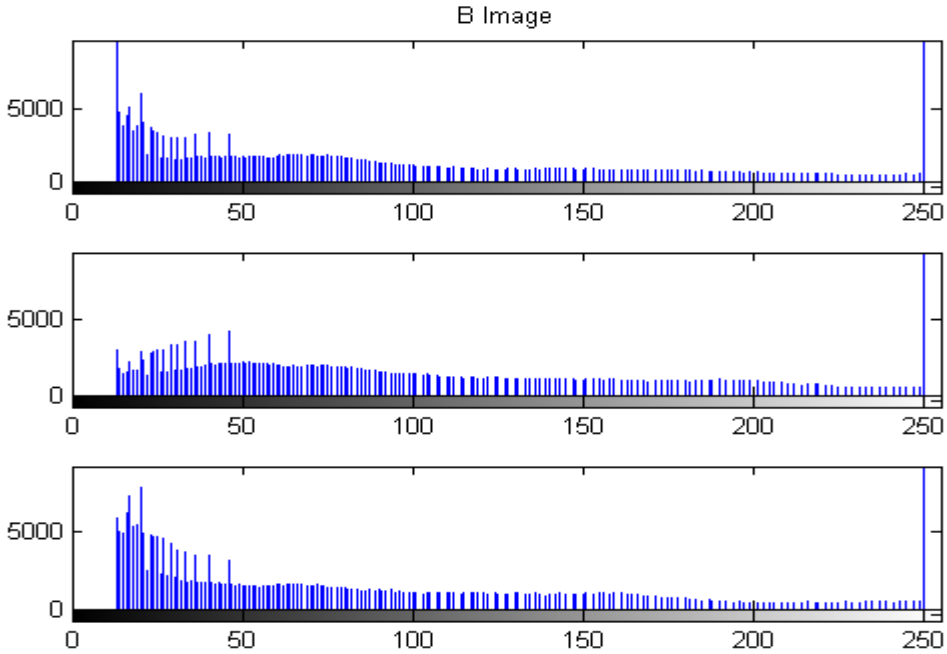


Figure 15. Histogram of psue after adjustment.

Figure 16. Image after adjusted.

Function imadjust is the basic tool for intensity transformations of grey-scale images.
After the brightness has been adjusted, the image in figure 16 looks better, there is no white veil for the image which makes the image suck.
Then I want to use histogram equalization.

```
clc;close all;clear all;
I = imread('psue.jpg');
Ir = I(:, :, 1);Ig = I(:, :, 2);Ib = I(:, :, 3);
Gr=histeq(Ir,256);        %histogram equalization
Gg=histeq(Ig,256);
Gb=histeq(Ib,256);
figure,
subplot(311),imhist(Gr),title('Histo of Histeq');
subplot(312),imhist(Gr);subplot(313),imhist(Gr);
G=cat(3,Gr,Gg,Gb);
figure,imshow(G);title('Histeq');
imwrite(G,'Histeq.jpg');
```

Figure 17. Histeq equalization.

It seems like this equalization doesn't work well. The color turns to bad, worse then the image get by Function imadjust. So later I will use the adjusted image to do more jobs.

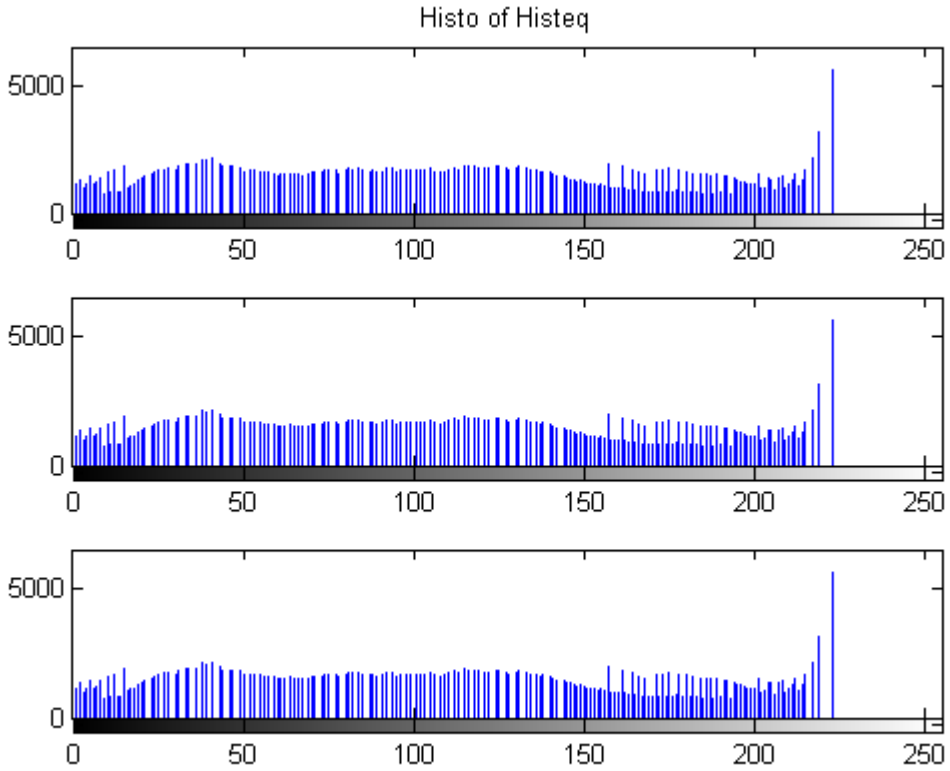


Figure 18.Histogram of Histeq

There is less high brightness and more low brightness. It should work but some unexpected happens. The image after equalization looks bad.

Here are the commands for changing background:

```
clc;close all;clear all;
O=imread('Adjust.jpg');        %Get a image from local storage
S=imread('bluesky2.jpg');
I=imread('psue.jpg');
A1=I(:,:,1);A2=I(:,:,2);A3=I(:,:,3);%Get each color plane
O1=O(:,:,1);O2=O(:,:,2);O3=O(:,:,3);%Get each color plane
[six,siy]=size(A1);
M=zeros(six,siy);        %create a new all 0 matrix.
%% build a control matrix
for i=1:six
    for j=1:siy
        if( (i-1.8333*j+518.32>0)&&(i+2.0625*j-689.375>0) ) % see figure 27
```

```
                M(i,j)=1;
            end
        end
    end
    figure,imshow(M);
    for i=1:six
        for j=1:siy
            if(M(i,j)==0)
                if ((A1(i,j)>240)&&(A2(i,j)>240)&&(A3(i,j)>240))
                                O1(i,j)=S(i,j,1);        O2(i,j)=S(i,j,2);
                        O3(i,j)=S(i,j,3);
                end
            end
        end
    end
    O=cat(3, O1, O2, O3);
    figure, imshow(O);
```

Figure19. Image with new background

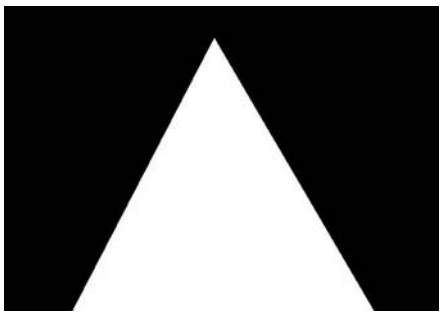


Figure 20. Controling image M

On the above process, M in figure 20 is used to control the process that copys the useful pixels to image Adjust. It contains only 0 and 1.When the value is 1, then the changing pixels is allowed.

The command A1=I(:,:, 1) help us to get the red plane image. As each pixel has two attribute: position and brightness. We can use I(i, j, 1) to locate a pixel and a= I(i, j, 1) to get the brightness value.

The command A=cat(3, A1, A2, A3) concatenate all color plane added together to represent the color image. As the color image is 3 dimensions, when we get each color plane image, we can stack the images again.

4.3 FFT2

First I use ideal low-pass filter to address another image, “cameraman. tif”, to show phenomenon of ringing. This TIF image is an original image cut off from a larger image from matlab help.

```
clc;close all;clear all;
I=imread('cameraman. tif');
A=I(:, :, 1);
figure (1), subplot(121), imshow(A); title('original');
[sizex, sizey]=size(A);
A2 = im2double(A); A22=fft2(A2); A12=fftshift(A22);
Figure(2), subplot(121), imshow(log(abs(A12)-1));title(‘  spectrum’  );
%% bulid an ideal low-pass filter with Butterworth low-pass filter

D0_lp = 25; % radius of the low-pass filter ring
n = 8;          % filter order
for i=1:sizex
```

```
        for j=1:sizey
            Dlp2 = (i-1-sizex/2)^2 + ((j-1-sizey/2)^2);% Dlp squared
            d2 = 1/exp( (sqrt(Dlp2)/D0_lp)^(2*n) ); % Butterworth low-pass filter
            if d2>0
                He(i, j)=1; % in equation 8 within D(u,v) ,the value is 1
            else
                He(i, j)=d2;    % in equation 8 outside D(u,v) ,the value is 0
            end
        end
    end
    figure(2), subplot(122), imshow(He), title('ILFilter');
    %%
    A122=A12. *im2double(He);
    A124=ifft2(ifftshift(A122));
    figure(1), subplot(122), imshow(A124); title('ring image');
```

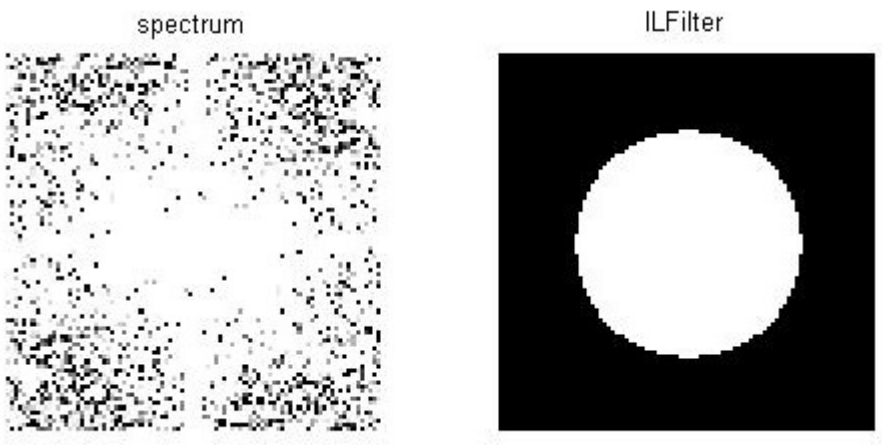


Figure 21. Spectrum of image and an ILFP image.

The spectrum diagram in figure 21 tells me that the original picture is not that good. Because the image has long edges where the pixel value change from low to high, even though I use command :log(abs(A12)-1), the spectrum is still almost white. The ILFilter image consists of 0 and 1. There is sharp jump from 0 to 1.

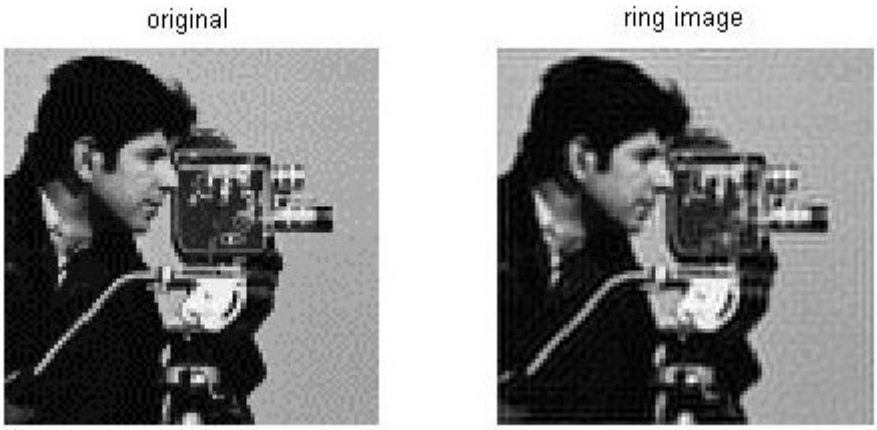


Figure 22. Ringing phenomenon.

Then let us see the effect ringing. From the ring image in figure 22 we can clearly see some strips are around the sleeve and the camera.

Because after changing the background, there are a lot of sharp edges. Then I can use Butterworth low-pass filter to smooth these edges in figure 19.

Here is the code:

```
clc;close all; clear all;
I= imread('Background.jpg');%use the image after imadjust function.
Ir = I(:, :, 1);Ig = I(:, :, 2);Ib = I(:, :, 3);
figure,
subplot(131), imshow(Ir, []);title('R');
subplot(132), imshow(Ig, []);title('G');
subplot(133), imshow(Ib, []);title('B');
[sizex, sizey]=size(Ir);

Ir2 = im2double(Ir); %double type complex values
Ir22=fft2(Ir2); %do fft2 transform
Ir12=fftshift(Ir22); %    Help to shift the values from four corners of the image to
the center

Ig2 = im2double(Ig); Ig22=fft2(Ig2); Ig12=fftshift(Ig22);
Ib2 = im2double(Ib); Ib22=fft2(Ib2); Ib12=fftshift(Ib22);
%%
```

```
Ir121=log(abs(Ir12)+2); Ig121=log(abs(Ig12)+2); Ib121=log(abs(Ib12)+2);
%after fft2, the values change to complex double, we use abs function to get a
double number.
figure,
subplot(131), imshow(Ir121,[]);title('R');
subplot(132), imshow(Ig121,[]);title('G');
subplot(133), imshow(Ib121,[]);title('B');

%%
D0_lp =149; % radius of the low-pass filter ring
n =4;          % filter order
for i=1:sizex
    for j=1:sizey
        Dlp2 = (i-1-sizex/2)^2 + ((j-1-sizey/2)^2)/1.96;% Dlp squared
        d2 = 1/(1+((sqrt(Dlp2))/(D0_lp))^(2*n));    % Butterworth low-pass filter
        H2(i,j)=d2;
    end
end
%%
for i=1:sizex
    for j=1:sizey
        Dlp3 = (i-1-sizex/2)^2 + ((j-1-sizey/2)^2)/1.49;% Dlp squared
        d3 = 1/(1+((sqrt(Dlp3))/(D0_lp))^(2*n)); % Butterworth low-pass filter
        H3(i,j)=d3;
    end
end
%%
for i=1:sizex
    for j=1:sizey
        Dlp4 = (i-1-sizex/2)^2 + ((j-1-sizey/2)^2)/1.96;% Dlp squared
        d4 = 1/(1+((sqrt(Dlp4))/(D0_lp))^(2*n)); % Butterworth low-pass filter
        H4(i,j)=d4;
    end
end
```

```
figure,
subplot(131), imshow(H2), title('H2');subplot(132), imshow(H3), title('H3');
subplot(133), imshow(H4), title('H4');

Ir122=Ir12.*im2double(H2); % address low-pass filter
Ir124=ifft2(ifftshift(Ir122));
  %shift the values in the image center back to four corners
Ig122=Ig12.*im2double(H3);Ig124=ifft2(ifftshift(Ig122));

Ib122=Ib12.*im2double(H4);Ib124=ifft2(ifftshift(Ib122));

Irh125=im2uint8(Ir124);Igh125=im2uint8(Ig124);Ibh125=im2uint8(Ib124);

J = cat(3, Irh125, Igh125, Ibh125);
figure, imshow(J);title('Cat Image');
imwrite(J,'BBter.jpg');
```

Figure 23. Three plane of psue image.

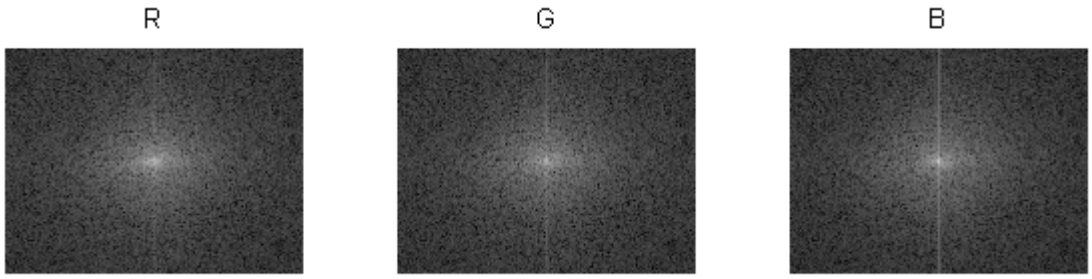


Figure 24. Spectrum of three plane

From figure 24, we can see that the Spectrum R, G and B are not same. Then it should be make different filter for them.

Figure 25. Designed Filter for the spectrum.

According to the rule of low-pass filter, the white area in the center of the spectrum image should be kept because different color channels have different spectrums, different filters that correspond to the spectrum is necessary. Then I come to the idea that the equation of an ellipse: $\left(\frac{x}{a}\right)^2 + \left(\frac{y}{b}\right)^2 = 1.$ （$a > b > 0$）(12).

We can set the value of a and b, get different shape of ellipses. Because the three filters have different white area in the center.

For example:

Dlp2 = (i-1-sizex/2)^2 + ((j-1-sizey/2)^2)/1. 96, in which Dlp2 is stand for u^2+v^2 in equation (9), I made v^2 divided by 1.96 so that the filter shape will change to a ellipse. Then the filter will fit the Spectrum better.

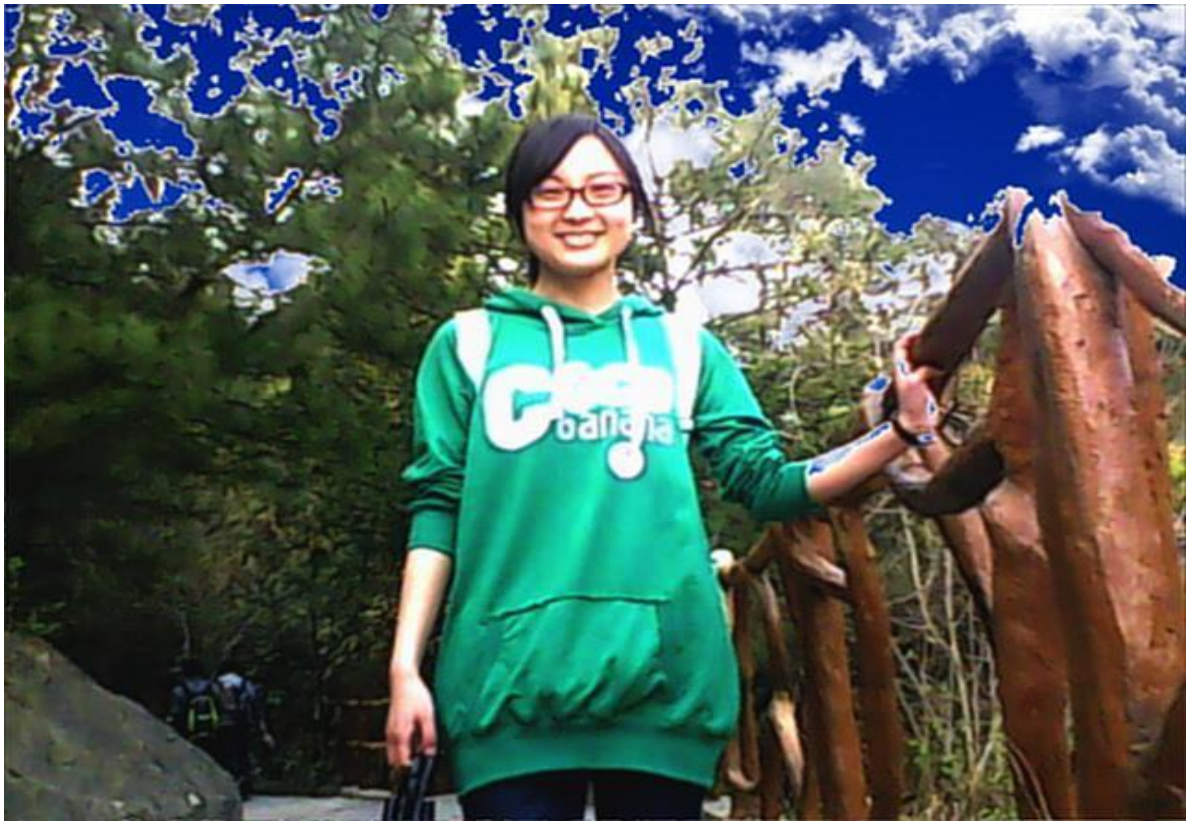


Figure 26. Cat image after low-pass filter.

Well, it looks like the filter does a good job to smooth the image. Some little dots around the edges are removed. Though it can evidently see that the blue sky is not real, but compared to white color background. It is better.

4.2 Adjusting colors

First I want to show an image. If we see the picture as a matrix, then at the upper left corner the coordinate is (0, 0). These lines help me to segment the image. And the two slash in the same length is used in page 20 to help create the control image so that the pixels under the triangle will not be changed.

Figure27. Puse image with marks.

Then I want segments two segments of the image:

```
clc;close all; clear all;
N= imread('BBter.jpg');
NB=imcrop(N, [210 160 219 299]);
```

```
imwrite(NB, ’ NB. jpg’ );
NC=imcrop(N, [430 0 209 159]);
imwrite(NC, ’ NC. jpg’ );
figure(1),
subplot(121), imshow(NB), title('NB');
subplot(122), imshow(NC), title('NC');
```

Figure 28. Two segments of the adjusted image BBter.

Then come to image NC, I want to amend some curing fence edge at the right.

```
clc;close all; clear all;
NC= imread('BBter. jpg');     %   image NC shown in figure 28.
M=zeros(159,210); % create a matrix

for i=105:159
    for j=110:210
        if(i-1.1*j+60>=0)   % help   to create the control image M
            if(j<130)
                NC(i, j, 1)=NC(i, 160, 1);
                NC(i, j, 2)=NC(i, 160, 2);
                NC(i, j, 3)=NC(i, 160, 3);
                M(i,j)=1;
            else
                NC(i, j, 1)=round(NC(i, 160, 1)*0.8+NC(i, j, 1)*0.2);
```

```
                    NC(i, j, 2)=round(NC(i, 160, 2)*0.8+NC(i, j, 2)*0.2);
                    NC(i, j, 3)=round(NC(i, 160, 3)*0.8+NC(i, j, 2)*0.2);
                    M(i,j)=1;
                end
            end
        end
    end
     figure(1),
    subplot(121), imshow(M), title('MCONTROL');
    subplot(122), imshow(NC), title('NC')
```

Here is the result:

Figure 29: Control image M and Amended Fence

Compared the image N in figure 28, the NC image in Figure 29 is better.

Then I get a part of IMG.JPG image. It is an image of the same person as shown in figure 19.

```
J= imread('IMG. jpg');
JB=imcrop(J, [240 180 189 269]);
%imwrite(JB, ' JB, jpg' );
figure, imshow(JB), title('JB');
```

Here are the two images I have added some marks on them:

Fiture30. Image NB with mark

Figure31. image JB with mark

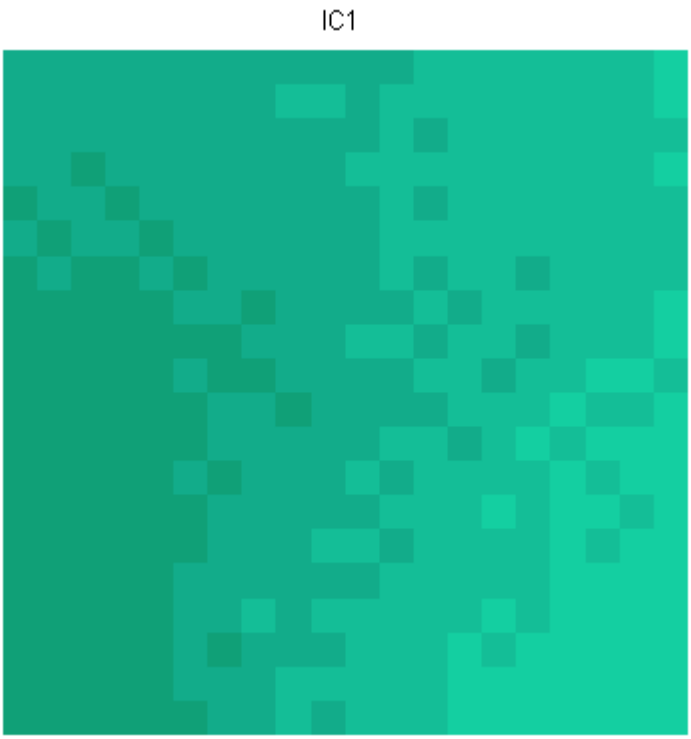
Now I want to use something about Indexed image:

```
IC=imcrop(NB, [80 120 19 19]);
JC=imcrop(JB, [85 140 19 19]);
figure(2),
subplot(121), imshow(IC), title('IC');subplot(122), imshow(JC), title('JC');
[IC1, map3] = rgb2ind(IC, 4);
figure, image(IC1);colormap(map3), title('IC1');
axis off
axis image
[JC1, map4] = rgb2ind(JC, 4); % create a index image with 4 indexed color
figure, image(JC1);colormap(map4), title('JC1');
axis off
axis image
mapi=im2uint8(map3);
mapj=im2uint8(map4);
[sizex, sizey]=size(NB(:, :, 1));
  %
for i=1:sizex
  for j=1:sizey
     if((abs(NB(i, j,1)-15)<3)&&(abs(NB(i, j,2)-158)<3)&&(abs(NB(i, j,3)-122)<3))
                    NB(i, j, 1)=23;
```

```
                    NB(i, j, 2)=105;
                    NB(i, j, 3)=76;
        elseif((abs(NB(i, j,1)-18)<3)&&(abs(NB(i, j,2)-191)<3)&&(abs(NB(i, j,3)-153)<3))
                    NB(i, j, 1)=30;
                    NB(i, j, 2)=126;
                    NB(i, j, 3)=89;
        elseif((abs(NB(i, j,1)-11)<3)&&(abs(NB(i, j,2)-175)<3)&&(abs(NB(i, j,3)-137)<3))
                    NB(i, j, 1)=34;
                    NB(i, j, 2)=138;
                    NB(i, j, 3)=99;
       elseif((abs(NB(i, j, 1)-20)<3)&&(abs(NB(i, j,2)-207)<3)&&(abs(NB(i, j,3)-168)<3))
                    NB(i, j, 1)=26;
                    NB(i, j, 2)=112;
                    NB(i, j, 3)=82;
                end
            end
        end
        figure, imshow(IB);
```

Figure 32. Segments from image NB and image JB.

From figure 32, we can see that in a 20x20 image matrix it can show more many kinds of colors. Most of them cannot be distinguished from each other.



Fifure33. IC1 indexed image.

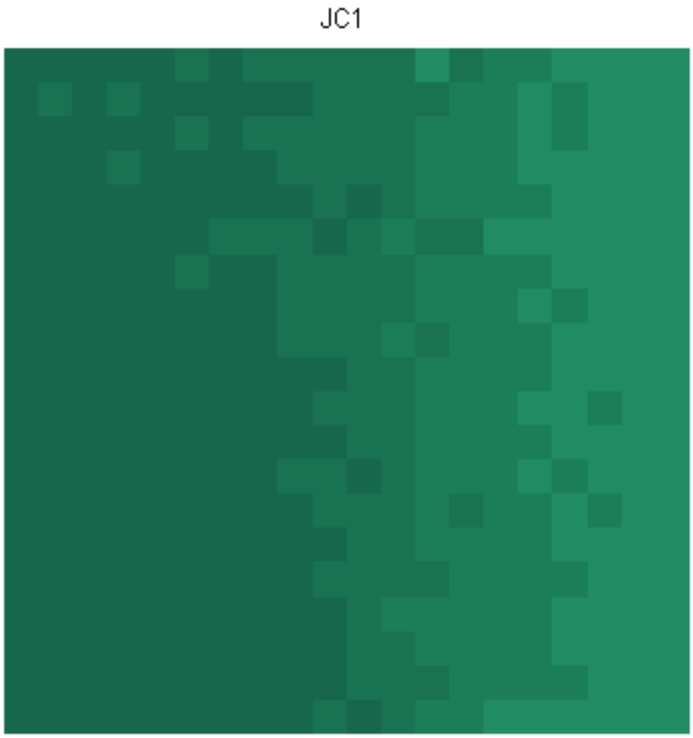


Figure 34. JC1 indexed image

From figure 33 and 34, we can evidently see 4 kinds of colors. The idea is that because the number of colors is small, we can easily change one of them or all of them eatly.

mapi

mapi <4x3 uint8>

	1	2	3	4
1	15	158	122	
2	18	191	153	
3	17	175	137	
4	20	207	168	

mapj

mapj <4x3 uint8>

	1	2	3	4
1	23	105	76	
2	30	126	89	
3	34	138	99	
4	26	112	82	

Figure 35. Maps from the two indexed image.

I try to use the mapj from Indexed image JC to enhance the indexed image IC. But it doesn't work well. Some more details should be considered. I believe this is a feasible way though. Here is the result which I got.

Figure 36. Colored changed image.

From the figure 36, we can see that also the pants have turn to green. The thing is that we can see at least three colors of the coat. It means that the way use indexed image to enhance the color is plausible.

5 CONCLUSION

Because of the problem related to the camera and the user, the image may have some issues with explosion.

After some any analyse, I do some compute, then I use function adjust to adjust the intensity level between 40 to 240, then the image looks better, less high intensity components move the white veil of the image.

Then I added a new background to the adjusted image. I made the original image to help me decide which pixels should be changed by using 240 as the down bound, above which the values of each pixel is copied by new values.

After the background has changed, I used a low pass filter to help me smooth the image so that those independent dots or sharp edges from white to blue were removed. During that process I used ellipse shape filter, it worked well. Then the image looks better, it is colorflul and the brightness is moderate.

Lastly, I segmented the image to enhance them separately. I amended the edge of the fence. I copied the pixels at the bottom of the segment of the image as it is more real and easy to manage. Then I did some experiments with indexed image to change the color of the coat.

BIBLIOGRAPHY

Gonzales, Rafael C. – Woods, Richard E 2002. Digital Image Processing. Second Edition. United States of America. Prentice Hall Pu-blishing.

Gonzales, Rafael C. – Woods, Richard E. – Eddins, Steven L 2004. Digital Image Processing using Matlab. United States of America. Gatesmark Pu-blishing.

Electronic sources

JPEG. http:// en. wikipedia. org/wiki/JPEG#Sample_photographs

Histogram Equalizationdetials.
http://fourier.eng.hmc.edu/e161/lectures/HistogramEqualization.pdf

FFT Details In Matlab.http://www. mathworks. se/help/matlab/ref/fft2.