# 实验二：空间域图像增强

姓名：鲁国锐 学号：17020021031 专业年级：电子信息科学与技术专业2017级 日期：2020年03月27号

**实验目标：**

1. 熟悉各种空间域滤波原理及操作。
2. 使用MATLAB（或Python）实现相应的滤波操作。
3. 完成给定的实验任务（见实验要求）。

**实验要求：**

1. 用Matlab（或Python）写一段程序，针对提供的图片IMG\_2546.JPG，实现：
2. 查看直方图
3. 取反，再查看直方图
4. 用直方图均衡，再查看直方图
5. 通过旋转、切割，仅保留“爱丁堡花园”部分
6. 针对图像100\_3228.JPG，使用图像增强的方法使图像的效果好一点，并对比增强前后的直方图变化。
7. 人脸1.jpg、2.jpg、3.jpg、4.jpg进行滤波等操作实现类似美图秀秀磨皮功能，并对比磨皮前后直方图变化。

**实验步骤（可以附加实验代码）：**

**任务一**

**思路：见实验要求1，其中取反操作为用255减去像素值**

**Python：**

from PIL import Image  
from pylab import \*  
import matplotlib.pyplot as plt  
  
def inverse0(im):  
 #im = array(im) # 由于可能对im1直接进行运算，对整型的像  
 # 素数据的除运算，会导致小数丢失。故需要  
 # 增加'f'option  
  
 for i in range(len(im)):  
 for j in range(len(im[i])):  
 for c in range(len(im[i][j])):  
 im[i][j][c] = 255 - im[i][j][c]  
  
 return im  
  
  
def histeq(im, nbr\_bins=256):  
 *""" 对一幅灰度图像进行直方图均衡化"""* #im = array(im, 'f') # 由于可能对im1直接进行运算，对整型的像  
 # 素数据的除运算，会导致小数丢失。故需要  
 # 增加'f'option  
 # 计算图像的直方图  
 imhist, bins = histogram(im.flatten(), nbr\_bins, normed=True)  
 cdf = imhist.cumsum() # cumulative distribution function  
 cdf = 255 \* cdf / cdf[-1] # 归一化  
  
 # 使用累积分布函数的线性插值，计算新的像素值  
 im2 = interp(im.flatten(), bins[:-1], cdf)  
  
 return im2.reshape(im.shape), cdf  
  
  
pil\_im = array(Image.open('100\_3228.JPG'))  
print(pil\_im.shape)  
plt.figure('1')  
  
plt.subplot(321)  
plt.imshow(pil\_im)  
plt.subplot(322)  
hist(pil\_im.flatten(), 128)  
  
im\_inverse = inverse0(pil\_im.copy())  
#im\_inverse = Image.fromarray(uint8(im\_inverse))  
plt.subplot(323)  
plt.imshow(im\_inverse)  
plt.subplot(324)  
hist(im\_inverse.flatten(), 128)  
  
  
img, cdf = histeq(pil\_im)  
  
print(type(img.shape))  
print(cdf.shape)  
#img = img/255  
img = Image.fromarray(uint8(img))  
plt.subplot(325)  
plt.imshow(img)  
plt.subplot(326)  
hist(cdf, 128)  
  
plt.show()

**MATLAB：**

figure(1)

% 原图

im = imread('IMG\_2546.JPG');

subplot(3, 2, 1)

imshow(im)

subplot(3, 2, 2)

imhist(im)

% 取反

%im\_bw = im2bw(im);

im\_inverse = im;

[row, col, channel] = size(im\_inverse)

for r = 1 : row

for c = 1 : col

for ch = 1 : channel

im\_inverse(r, c, ch) = 255 - im\_inverse(r, c, ch);

end

end

end

subplot(3, 2, 3)

imshow(im\_inverse)

subplot(3, 2, 4)

imhist(im\_inverse)

% 直方图均衡化

im\_eq = histeq(im);

subplot(3, 2, 5)

imshow(im\_eq)

subplot(3, 2, 6)

imhist(im\_eq)

saveas(1, 'C:\Users\Asus-\Desktop\数字图像\lecture\expericnce02\实验二 空间域图像增强\task1.jpg')

imwrite(im\_eq, 'C:\Users\Asus-\Desktop\数字图像\lecture\expericnce02\实验二 空间域图像增强\eq.jpg')

figure(2)

im\_rotate = imrotate(im, -15);

imshow(im\_rotate)

size(im\_rotate)

figure(3)

im\_crop = im\_rotate(700 : 900 , 600: 1200, :);

imshow(im\_crop);

**任务二**

**思路：对比各种操作之后，最终决定用分段幂次变换的方式：**

* 1. 打开一幅图片
  2. 遍历图像：
     1. 若三个通道的像素值都小于50，进行以1.2为底的幂次变换
     2. 若三个通道的像素值都小于100，进行以1.08为底的幂次变换
     3. 若三个通道的像素值都小于150，进行以1.05为底的幂次变换
     4. 若以上三个条件均不满足，不进行任何变换
  3. 保存图像
  4. 显示变换后的图像及其直方图

im=imread('100\_3228.JPG');

%im = double(im);

% 显示原图及其直方图

figure(1)

subplot(2, 1, 1)

imshow(im)

subplot(2, 1, 2)

imhist(im)

threshold1 = 50;

threshold2 = 100;

threshold3 = 150;

base1 = 1.2;

base2 = 1.08;

base3 = 1.05;

%base4 = 1.028;

im\_pow = double(im);

%im\_pow = min(200, power(1.2, im\_pow));

[rows, cols, channels] = size(im\_pow);

for row = 1 : rows

for col = 1 : cols

%for c = 1 : channels

if im\_pow(row, col, 1) < threshold1 && im\_pow(row, col, 2) < threshold1 && im\_pow(row, col, 3) < threshold1

im\_pow(row, col, 1) = min(255, power(base1, im\_pow(row, col, 1) ));

im\_pow(row, col, 2) = min(255, power(base1, im\_pow(row, col, 2) ));

im\_pow(row, col, 3) = min(255, power(base1, im\_pow(row, col, 3) ));

elseif im\_pow(row, col, 1) < threshold2 && im\_pow(row, col, 2) < threshold2 && im\_pow(row, col, 3) < threshold2

im\_pow(row, col, 1) = min(255, power(base2, im\_pow(row, col, 1) ));

im\_pow(row, col, 2) = min(255, power(base2, im\_pow(row, col, 2) ));

im\_pow(row, col, 3) = min(255, power(base2, im\_pow(row, col, 3) ));

elseif im\_pow(row, col, 1) < threshold3 && im\_pow(row, col, 2) < threshold3 && im\_pow(row, col, 3) < threshold3

im\_pow(row, col, 1) = min(255, power(base3, im\_pow(row, col, 1) ));

im\_pow(row, col, 2) = min(255, power(base3, im\_pow(row, col, 2) ));

im\_pow(row, col, 3) = min(255, power(base3, im\_pow(row, col, 3) ));

% else

% im\_pow(row, col, 1) = min(255, power(base4, im\_pow(row, col, 1) ));

% im\_pow(row, col, 2) = min(255, power(base4, im\_pow(row, col, 2) ));

% im\_pow(row, col, 3) = min(255, power(base4, im\_pow(row, col, 3) ));

end

%end

end

end

im\_pow = uint8(im\_pow);

imwrite(im\_pow, 'C:\Users\Asus-\Desktop\数字图像\lecture\expericnce02\实验二 空间域图像增强\p1\_2\_and\_1\_08\_and\_1\_05\_and\_1\_03.jpg');

figure(2)

subplot(2, 1, 1)

imshow(im\_pow)

subplot(2, 1, 2)

imhist(im\_pow)

**附：锐化方案代码（未采用）：**

%%

im=imread('100\_3228.JPG');

%im = double(im);

% 显示原图及其直方图

figure(1)

subplot(2, 1, 1)

imshow(im)

subplot(2, 1, 2)

imhist(im)

%h=[1,2,1;0,0,0;-1,-2,-1];%Sobel算子

h=[1,0,-1;2,0,-2;1,0,-1];

%h=[0,1,0;1,-4,1;0,1,0];%拉氏算子

im\_filted1=filter2(h,im(:, :, 1));

im\_filted2=filter2(h,im(:, :, 2));

im\_filted3=filter2(h,im(:, :, 3));

im\_filted = cat(3, im\_filted1, im\_filted2);

im\_filted = cat(3, im\_filted, im\_filted3);

% 显示滤波后的图像及其直方图

figure(2)

subplot(2, 1, 1)

imshow(im\_filted)

subplot(2, 1, 2)

imhist(im\_filted)

im\_res = im;

[row, col, channel] = size(im);

for i = 1 : row

for j = 1 : col

for c = 1 : channel

temp = uint16(im(i, j, c));

temp = temp + uint16(im\_filted(i, j, c));

im\_res(i, j, c) = temp;

end

end

end

h=[1,2,1;0,0,0;-1,-2,-1];

%h=[0,1,0;1,-4,1;0,1,0];%拉氏算子

im\_filted1=filter2(h,im(:, :, 1));

im\_filted2=filter2(h,im(:, :, 2));

im\_filted3=filter2(h,im(:, :, 3));

im\_filted = cat(3, im\_filted1, im\_filted2);

im\_filted = cat(3, im\_filted, im\_filted3);

for i = 1 : row

for j = 1 : col

for c = 1 : channel

temp = uint16(im\_res(i, j, c));

temp = temp + uint16(im\_filted(i, j, c));

im\_res(i, j, c) = temp;

end

end

end

im\_res = uint8(im\_res);

figure(6)

subplot(2, 1, 1)

imshow(im\_res)

imwrite(im\_res, 'C:\Users\Asus-\Desktop\数字图像\lecture\expericnce02\实验二 空间域图像增强\100\_3228\_enhanced.jpg')

subplot(2, 1, 2)

imhist(im\_res)

**任务三**

**思路：**

1. **打开图片**
2. **显示原图及其直方图**
3. **分别对三个通道进行均值滤波**
4. **将滤波后的三个通道拼接成RGB图像**
5. **显示均值滤波后的图像及其直方图，并保存图像**
6. **对原图进行中值滤波（可直接用medfilt3函数，不必对三个通道分别滤波）**
7. **显示中值滤波后的图像及其直方图，并保存图像**

im=imread('4.jpg');

%im = double(im);

% 显示原图及其直方图

figure(1)

subplot(2, 1, 1)

imshow(im)

subplot(2, 1, 2)

imhist(im)

%im = double(im);

%h=[1,1,1;1,1,1;1,1,1];

dim = 9;

h = ones(dim);

h = h / (dim.^2);

im\_filted1=conv2( im(:, :, 1), h, 'same');

im\_filted2=conv2( im(:, :, 2), h, 'same');

im\_filted3=conv2( im(:, :, 3), h, 'same');

im\_filted = cat(3, im\_filted1, im\_filted2);

im\_filted = cat(3, im\_filted, im\_filted3);

im\_filted = uint8(im\_filted);

% 显示滤波后的图像及其直方图

figure(2)

subplot(2, 1, 1)

imshow(im\_filted)

imwrite(im\_filted, './4junzhi.jpg')

subplot(2, 1, 2)

imhist(im\_filted)

im\_filted = medfilt3(im, [dim,dim,3]);

figure(3)

subplot(2, 1, 1)

imshow(im\_filted);

imwrite(im\_filted, './4medfilt3.jpg')

subplot(2, 1, 2)

imhist(im\_filted)

**实验结果：**

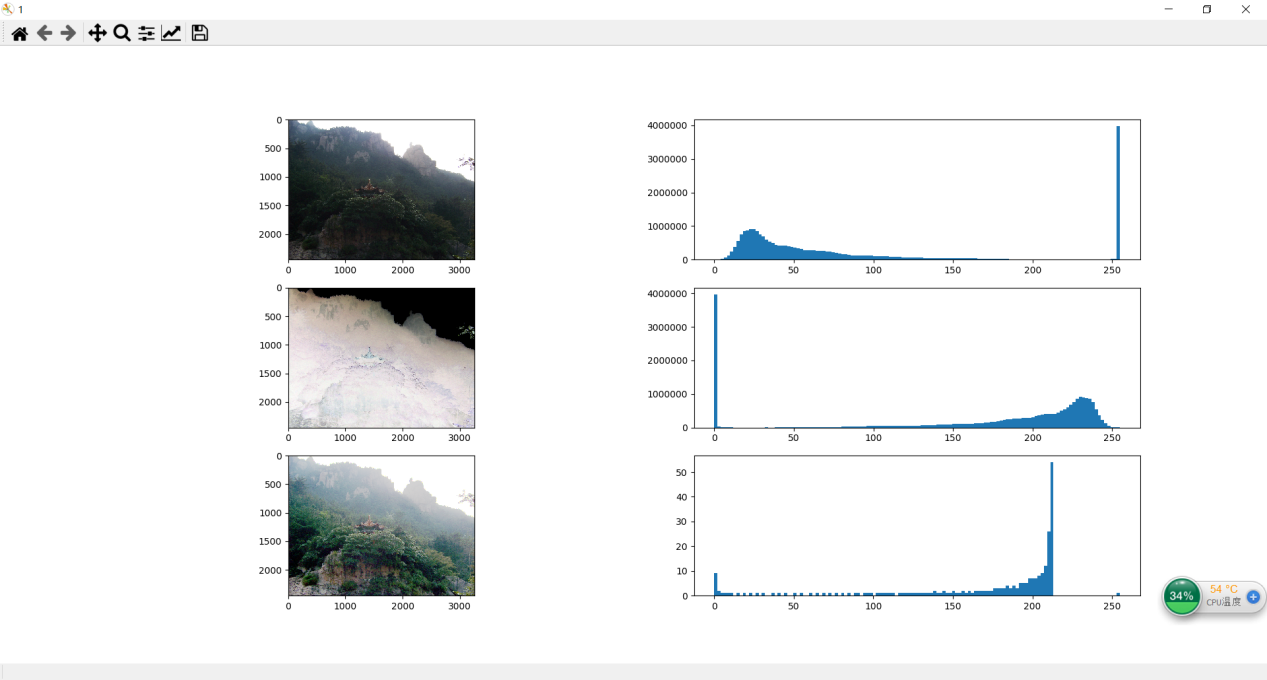


图 1实验一：Python版结果（前三步）

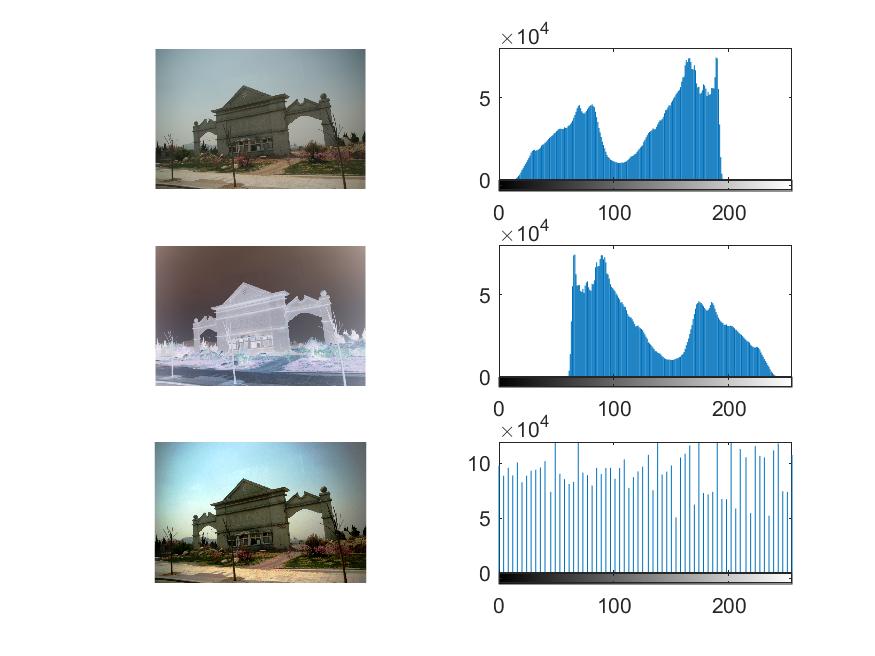
****

图 2实验一：MATLAB版结果（前三步：原图，取反结果，直方图均衡化结果）

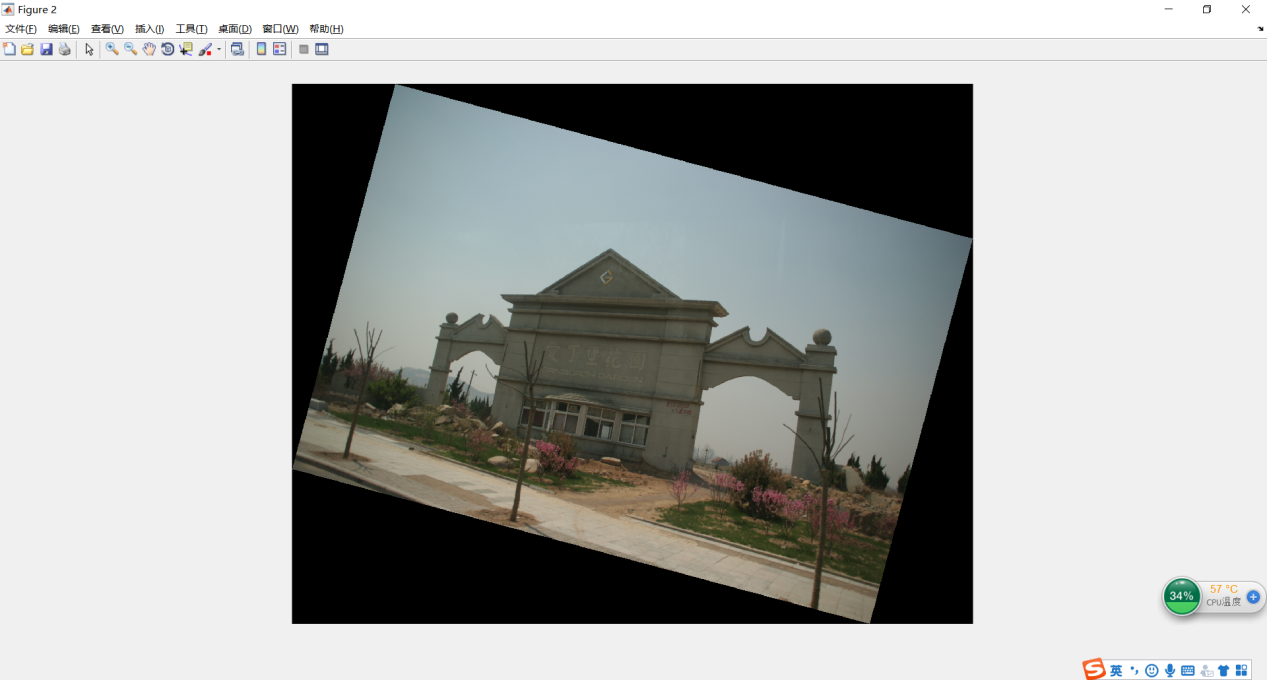


图 3实验一：旋转之后的图像



图 4 实验一：旋转裁剪后的图像

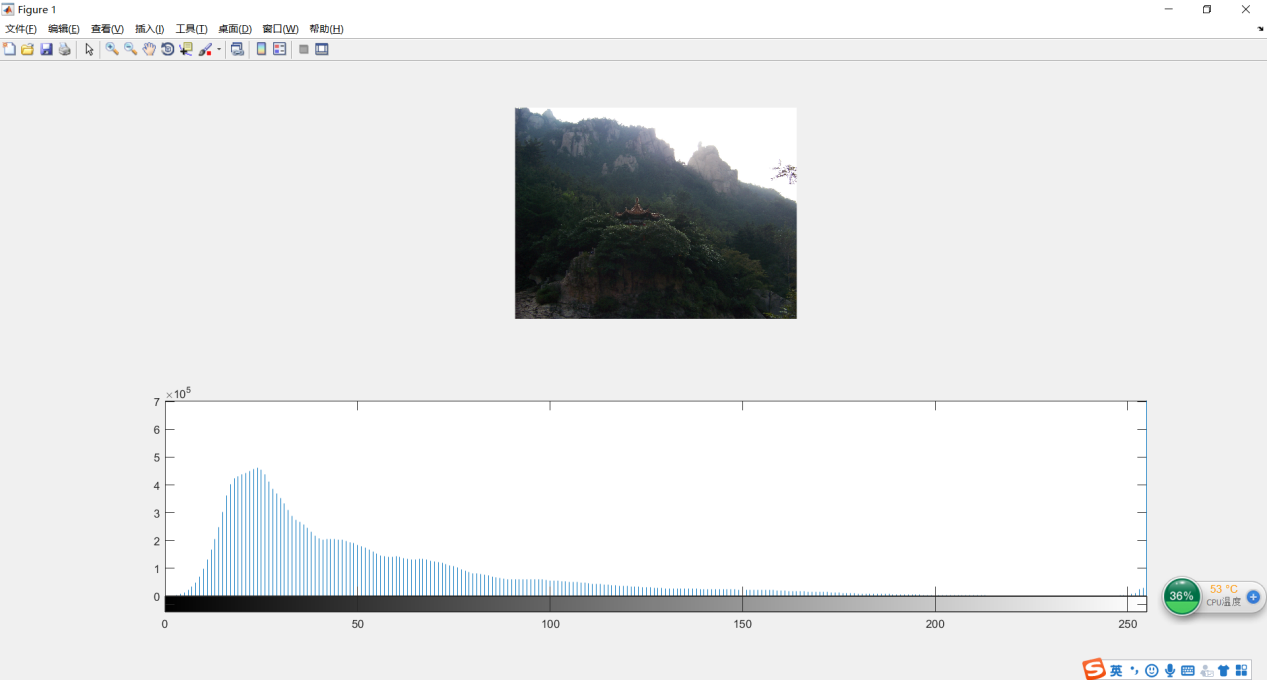


图 5 实验二：原图及其直方图



图 6 实验二：变换后的图像

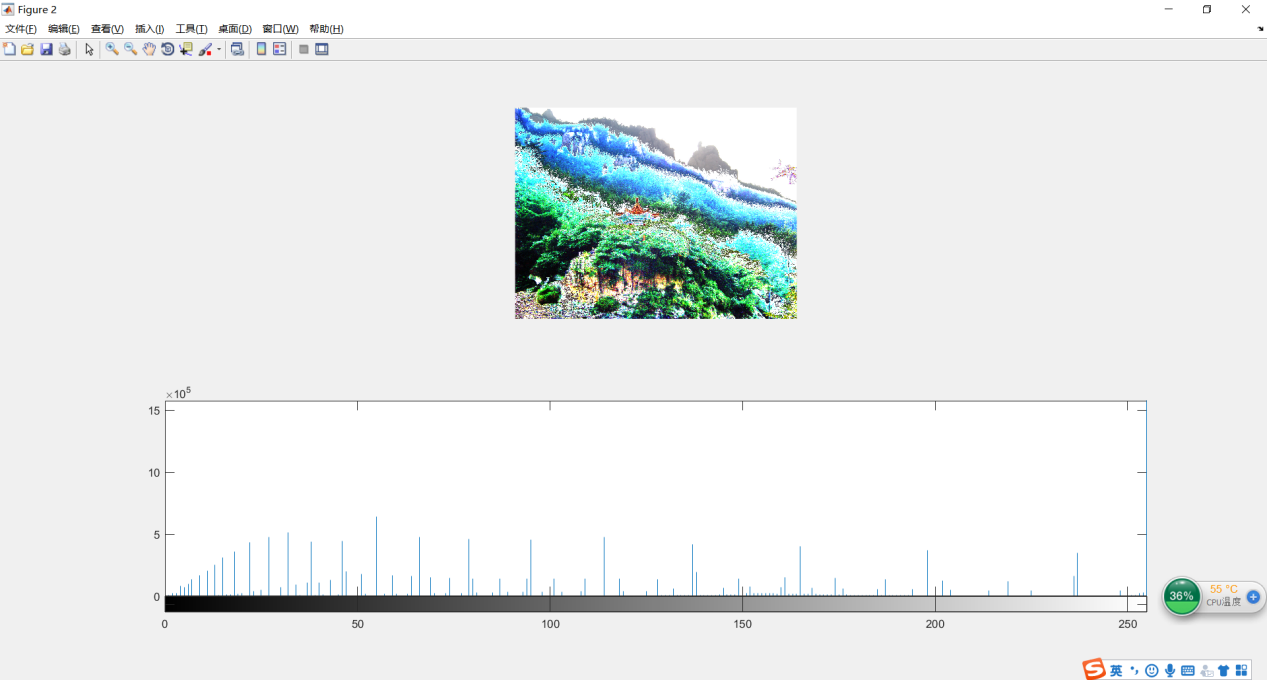


图 7 实验二：变换后的图像及其直方图



图 8 实验二：直方图均衡化后结果

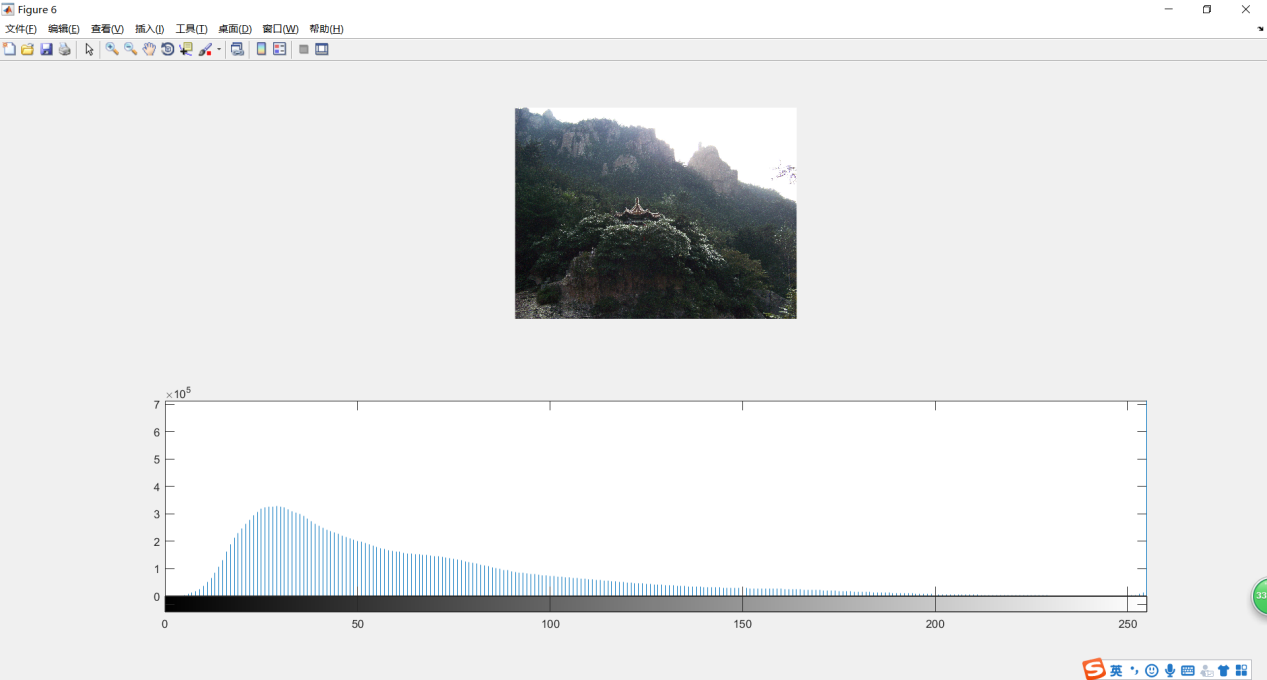


图 9 实验二：Sobel算子锐化后的图像及其直方图

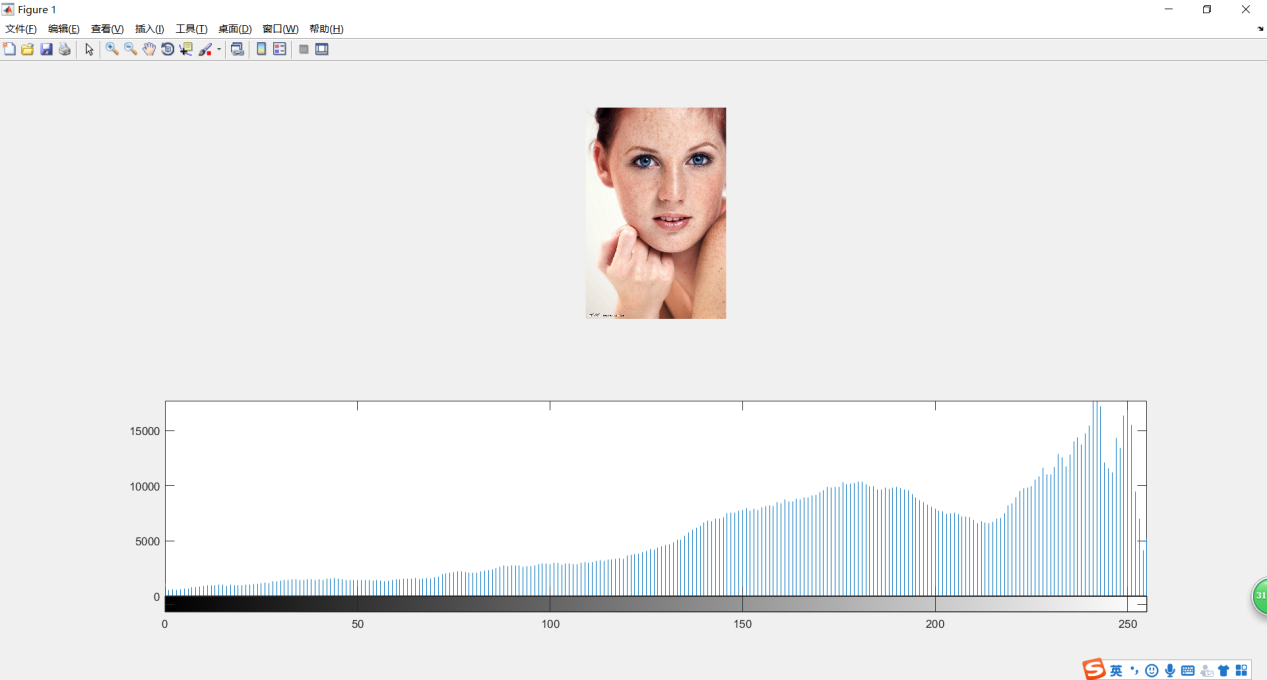


图 10 实验三：4.jpg原图及其直方图

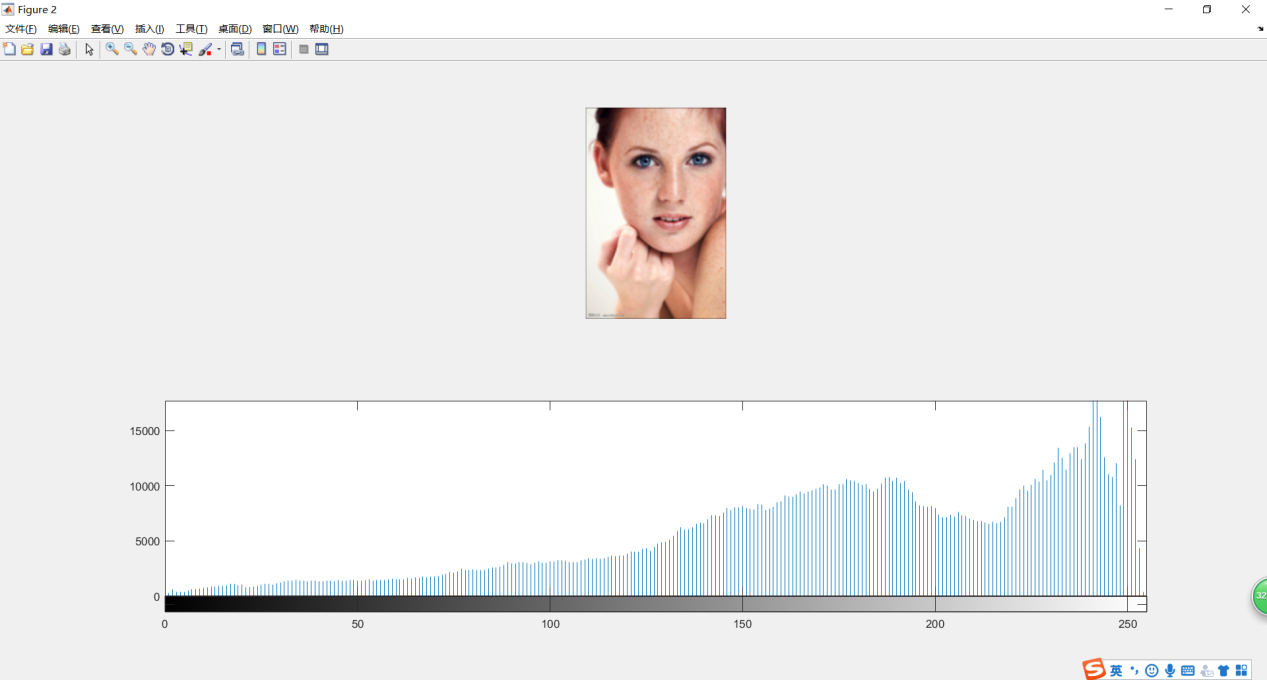


图 11 实验三：4.jpg均值滤波后图像及其直方图

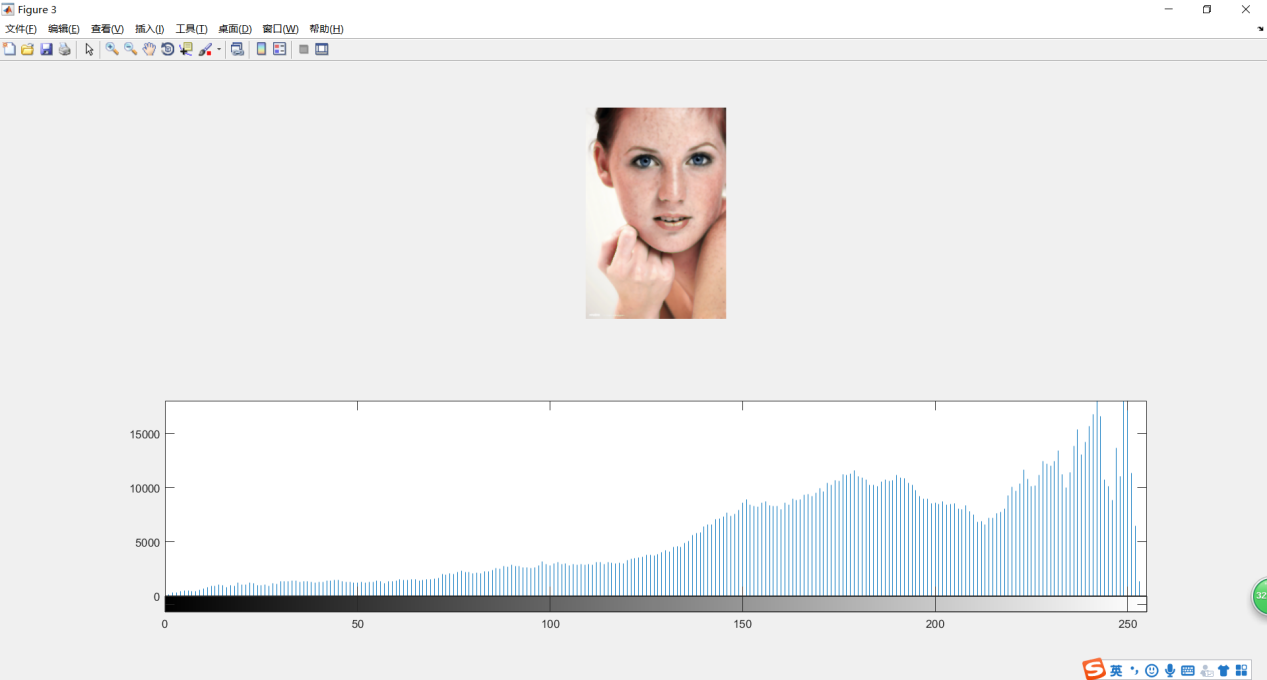


图 12 实验三：4.jpg中值滤波后图像及其直方图

**实验总结：**

本次实验共分三部分。第一部分仍然主要是熟悉图像处理的基本操纵，包括读取图像，查看直方图，旋转，裁剪等，同时也熟悉了取反和直方图均衡化这两个变换的操作流程。

第二部分相对而言是花时间最多的。对于这一部分，**其实用直方图均衡化就可以得到不错的效果，但这一方法在实验一中已经用过，故而在这里我们希望能找到其它的方法**。最初采取的方法是用Sobel算子或拉普拉斯算子对图像进行锐化，但得到的效果并不好（结果中附上了Sobel算子锐化后的结果，拉普拉斯算子的效果还要更差一点）。将资料里给的各种方法都试过一遍之后发现，幂次变换效果最好。但是若对整幅图像都采用相同的底数，会导致部分区域过亮或过暗，所以我们采取对像素值进行分段幂次变换的方法。对于不同取值范围的像素，我们采用不同底数的幂次变换：对于像素值本身较小的区域，可以用稍大一些的底数；而对于像素值本身就比较大的区域，则可将底数调小一些。另外，如果对于同一位置，三个通道的变换不一致，会导致部分区域（如森林墨绿色部分）颜色完全变成另一个样子（如墨绿色变为红色或蓝色），因此在对像素进行判断的时候要三个通道一起判断，变换也要一起变换，这样至少可以让颜色的性质不会有太剧烈的变化。

第三部分相较于第二部分反而要简单一点，直接用均值滤波和中值滤波即可。只是这里为了能得到彩色的处理结果，我们分别对三个通道进行滤波，再将这三个结果拼接成一个RGB图像。前面用Sobel算子和拉普拉斯算子锐化图像也是用的这样的方法。至于最终效果，均值滤波比中值滤波更好一些；滤波器选得大一些比小一些更好，但对于图像边缘的模糊也更严重。