**广州大学学生实验报告**

**开课学院及实验室：**计算机学院  **2020年 12月9日**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **学 院** | 计算机 | **年级/专业/班** | 网络193 | **姓名** | 赖杰伟 | **学号** | 1906200093 |
| **实验课程名称** | **数字图像处理** | | | | | **成绩** |  |
| **实验项目名称** | **数字图像处理大作业** | | | | | **指导老师** |  |

(\*\*\*（1）报告只能为文字和图片；（2）实验项目名称不能有任何改动；（3）必须填写正确的姓名和学号；否则会影响最后成绩的统计，后果自负\*\*\*)

**一、实验目的**

1. 掌握图像处理基本操作；按照既定格式书写实验报告。

**二、实验设备与平台**

1. 实验设备：计算机；

2. 平台：Windows/Linux操作系统。

**三、实验内容**

**(1)** 将原始图像 分解为 个子图像( 方向个，方向个)。如果 M=4，N=4. 则分解结果如下：

**原始图像 分解为4\*4个子图像**

**(2)** 图像经过分解之后得到个子图像，对每个子图像进行如下操作。（假设为任意一个子图像）

① 求出子图像的直方图，将中大于阈值T1的部分置为T1,小于阈值T2的部分置为0。即

If

；

If

;

其中 , , max(h)为h中的最大值

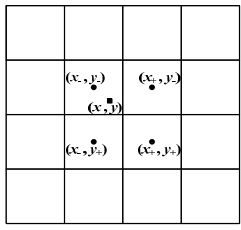
② 通过步骤 ① 得到新的直方图.求出每个灰度i 的概率密度. 然后对进行累加操作，得到原始图像中灰度级 映射后的灰度级



其中N=256， 为的整数部分。

③ 得到之后，将原始图像中灰度级为 的像素点替换为新的灰度级。所有的灰度级替换完毕得到新的增强后子图像。

**(3)** 插值操作：假设 为 *x*方向第 个， *y*方向第 个子图像，对非周边子图像 ()中的每个点进行如下插值操作，消除块效应。



假设为原始图像中任意一点，其灰度值为 （增强之前的灰度值）。, 是其周围4个子图像的中心点位置（如上图所示）。通过步骤2可以得到该四个子图像的四个灰度映射函数。

假设

则点处灰度级在插值后的灰度级变为：



其中



**实验要求：**

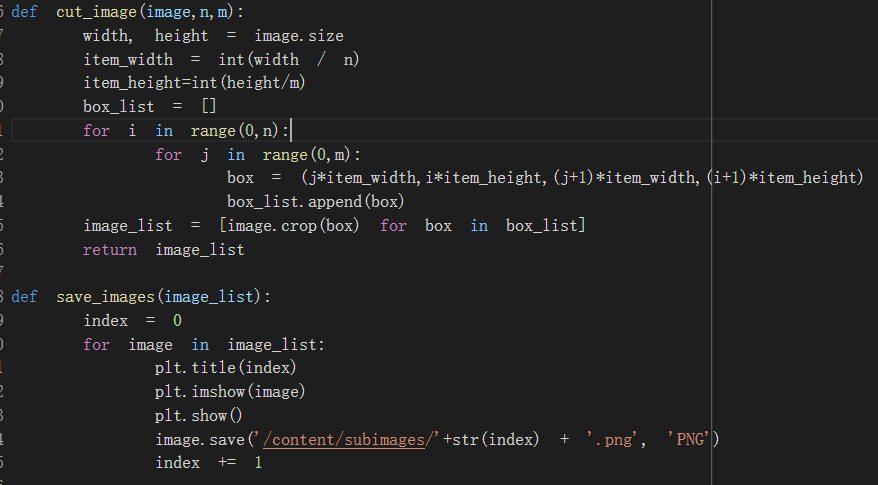
1. 使用Matlab / Python / C++ 语言设计；
2. 详细描述所使用的数据。把编写的程序代码和运行结果截图复制到题目后面的空白处。
3. 实验报告必要有效果截图

**四、实验记录和结果分析**

**（1）第一小题是对图像进行分割，将图片分割成n\*m张子图。**

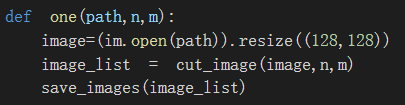
**在这里通过定义一个cut\_image函数实现对图像的分割，save\_images函数实现对子图的保存。首先建立子图的对角线坐标列表box\_list来确定每个子图的位置，在将坐标点传入image.crop函数中实现box区域内子图的提取，最后按照子图提取的顺序（0到n\*m）进行命名保存。这里需要注意的是，为了让得到的子图形状大小一致，在进行分割之前先将图像resize成128\*128大小，再根据认为设定的n，m值确定子图的长宽（转化成int型数据），此部分在one函数中实现。**

**Cut\_image,save\_images函数实现代码如下：**



**调用one函数可以完成第一题中的图像分割功能**

**One函数实现代码如下：**



**实验效果如下：（设定n，m值为4）**

**原始图像：**



**实验效果：**

** **  ****

****   

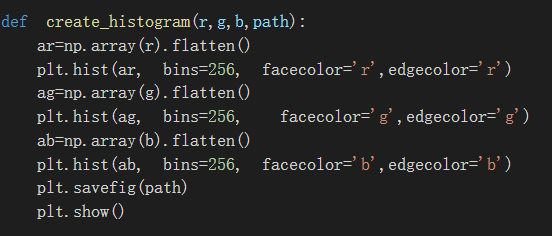
   

**（2）**

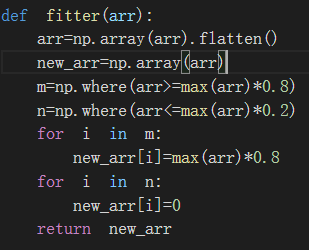
**首先定义一个create\_histogram函数进行图像直方图的生成，在后续的所有实验中，我们将图像提取出rgb三个通道的图片，然后对每个通道的图片都进行相应的处理。在create\_histogram函数中，我们传入rgb三个通道图片想以及直方图的保存路径，然后将图像转化成一维数据，利用PLT.his函数进行直方图绘画，三个通道对应的直方图填充对应的颜色，如r通道图片的直方图填充red颜色。**

**create\_histogram函数实现代码如下：**



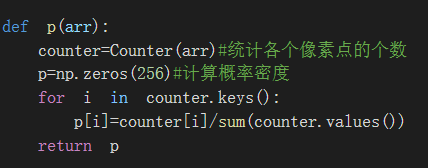
**通过fitter函数实现对原始图像中像素值的“截断”，fitter函数中，我们先构造一个new\_arr作为原始图像数组的副本，然后通过np.where筛选出原始图像中像素值大于T1和像素值小于T2的像素点下标，生成m，n两个列表分别代表两种目标像素值下标，然后对每个元素进行重置，返回处理后的数组new\_arr。**

**Fitter函数实现代码如下：**



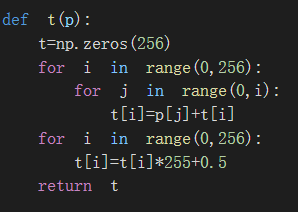
**我们通过定义p函数获取各个像素值的概率密度。通过collections中的Counter函数直接统计每个像素值出现的个数，然后利用循环计算各个像素值的密度返回密度函数p。**

**P函数实现代码如下：**



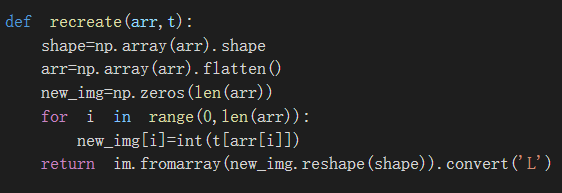
**我们通过定义t函数获取图像的灰度级函数。通过循环对像素的概率密度进行累加，在进行一次循环执行t=t\*255+0.5，获得每个像素点对应的灰度级，返回所有像素值的灰度级函数。**

**T函数实现代码如下：**



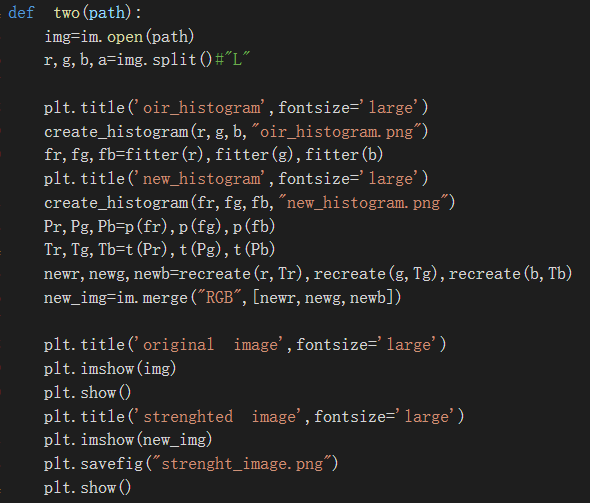
**通过定义recreate函数实现原始图像像素值的替代，从而生成增强后的图片。首先生成一个一维数组，通过循环语句将图像中每个点对应的像素值作为索引传入灰度级函数t中，将灰度级函数返回的数值作为新的像素值，从而得到新的图像数组，最后将得到的图像转化成灰度图像，便于后续对rgb三通道图像的合成。**

**Recreate函数实现代码如下：**



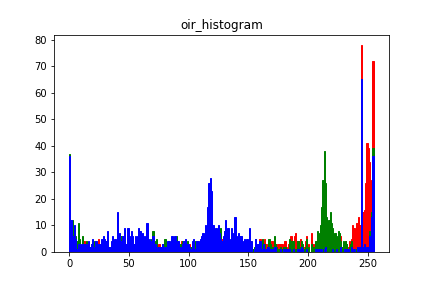
**通过调用two可以解决实验（2）中的问题，这里通过调用merge函数实现三通道图像的合并。**

**two函数实现代码如下：**

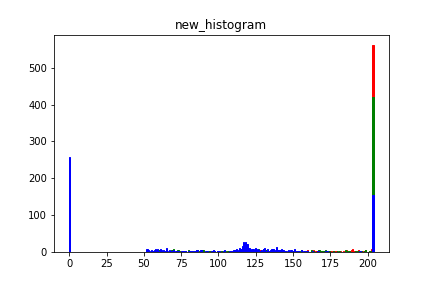


**实验效果如下：**

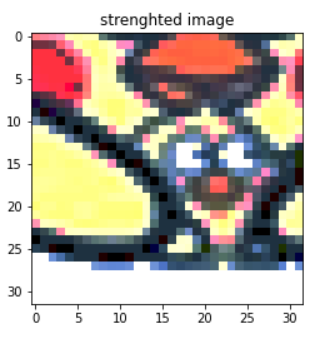
**原始图像直方图如下：**



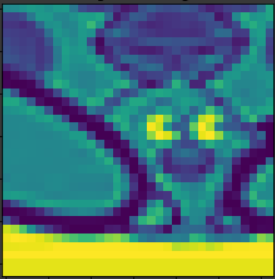
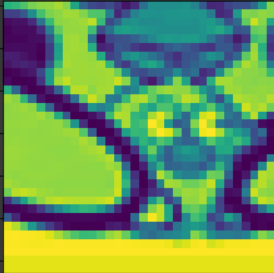
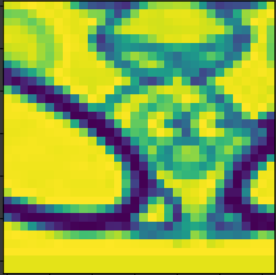
**增前后的图像直方图如下：**



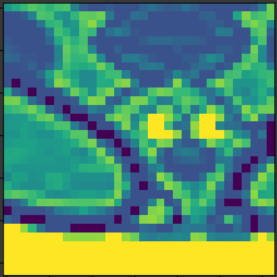
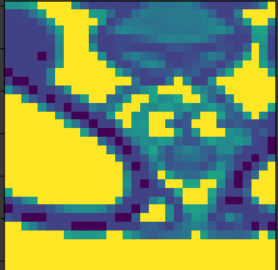
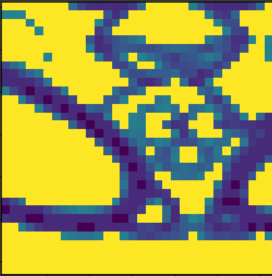
**原始图像： 增强后图像：**



**原始图像RGB图像：（从左到右依次为rgb）**



**增强图像RGB图像：（从左到右依次为rgb）**

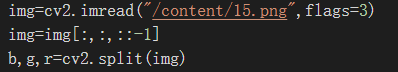


**另外，在这里我用了两个图像处理常用库实现了此部分的代码，上面展示的代码是由plt库实现的代码，下面部分是用opencv实现的代码。Opencv和plt在实现上有一些不同，首先是在图像的读取上，opencv读取后的图像会显示变成一些色彩奇怪的图片，如下图所示：**

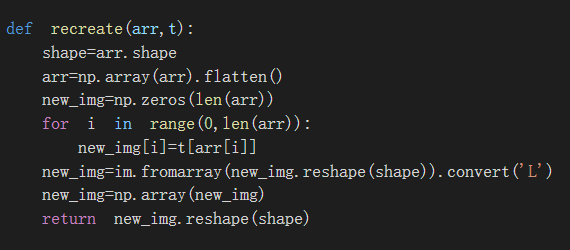


**因此要加上一些语句保证读取的图显示正常的；另外在opencv中RGB图像的通道顺序是不一样的，在plt中RGB图像的通道顺序为RGB（实际上在plt中图像会有四个通道，第四个通道代表透明度）而在opencv中通道顺序为BGR，因此在图像通道提取的合成时需要改变各个通道图像的顺序。**

**Opencv在图像读取和通道图像提取的实现代码如下：**



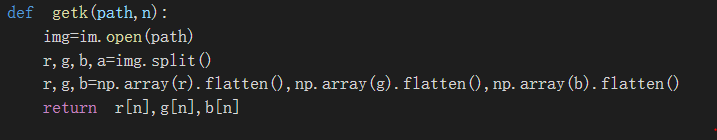
**另外，两个库的merge函数的参数也有所不同，plt中Merge函数第二个参数包含的rgb图片的数组列表，而opencv中merge函数中直接是rgb图片列表，因此在opencv实现方法中，在recreate函数汇总返回的是图片，如下所示：**



**（3）**

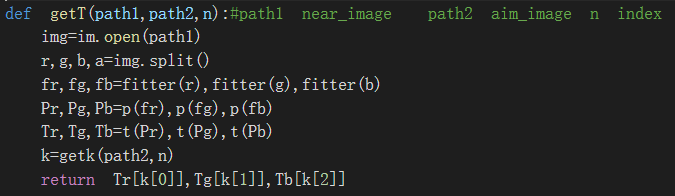
**为了在遍历过程中读取图片（x,y）图的像素值k，我们定义getk函数。通过图像的路径path读取图像，将图像提取出rgb三通道图像，将图像转化成一维数组，通过传入的下标n获得rgb图像对应的像素值，返回rgb三张图第n+1个元素的像素值列表。**

**Getk函数实现代码如下：**



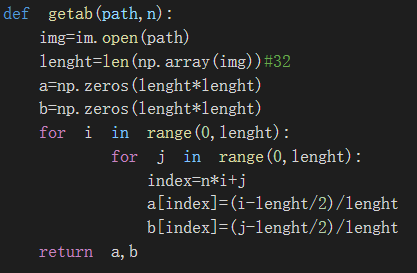
**通过getT函数分别获取图像RGB三个通道图像k像素值的灰度级。**

**通过依次调用fitter，p，t函数依次获取“截断”后的图像，像素值的概率密度和图像的灰度级，在通过将getk返回的灰度值获取该点对应的灰度级，返回该点对应的rgb三个通道的灰度级列表。**



**通过getab函数获得每个（x,y）对应的a，b的值。由于对于每张图片来说，每个(x,y)对应的a，b值都是固定的，所以a，b的值只需要计算一次即可，将a，b存储为列表形式，在需要时直接通过索引进行获取即可。**

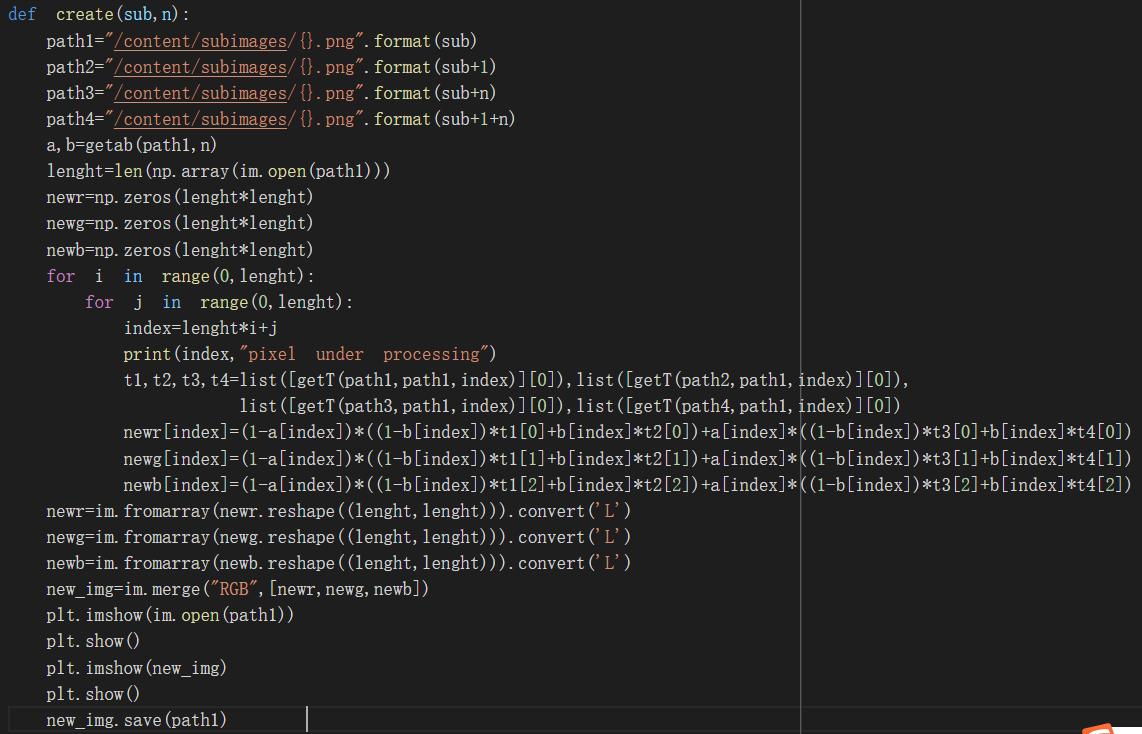
**Getab函数实现代码如下：**



**通过定义create函数实现对图像每个（x，y）进行替换。**

**在此函数中，我们传入要进行替换的图片的路径和n值（来源于n\*m）。通过图像的命名特征（+1，+n，+n+1）来读取周边的四张子图。通过np.zeros函数创建三个与子图长度相等的三个一维数组分别代表rgb三张图片；通过遍历用计算得来的值填充一维数组，最后将数组转化成灰度图像，再利用merge函数对rgb三个通道的图像进行合并，生成消除块效应后的子图。最后保存图像。**

**create函数实现代码如下：**



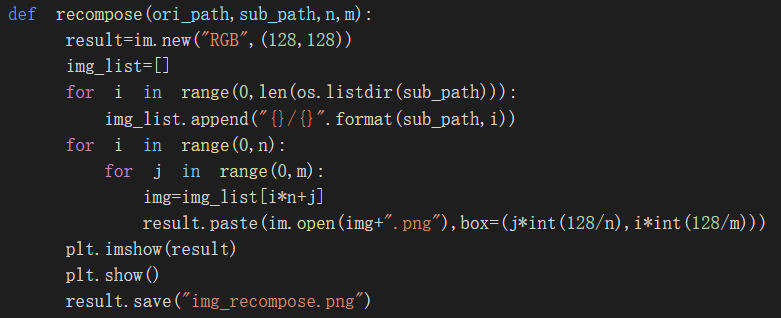
**实验效果如下：**

**原始图像： 消除块效应后的图像：**



**通过recompose函数对消除块效应过后的子图进行拼接。将所有子图以列表的形式读取到img\_list中，利用参数box确定子图拼接的位置，利用paste函数实现所有子图的拼接。**

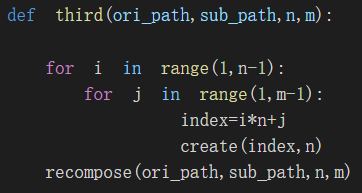
**Recompose函数实现代码如下：**



**通过third函数可以实现（3）的内容。**

**通过循环和调用create函数对非周边子图进行消除快效应，利用recompose函数对处理后的子图进行拼接生成新的图像并保存。**

**Third函数实现代码如下：**



**实验效果如下：**

**原始图像： 生成的图像：**



**实验完整代码如下：**

from PIL import Image as im

from matplotlib import pyplot as plt

from collections import Counter

import numpy as np

import os

def cut\_image(image,n,m):

    width, height = image.size

    item\_width = int(width / n)

    item\_height=int(height/m)

    box\_list = []

    for i in range(0,n):

        for j in range(0,m):

            box = (j\*item\_width,i\*item\_height,(j+1)\*item\_width,(i+1)\*item\_height)

            box\_list.append(box)

    image\_list = [image.crop(box) for box in box\_list]

    return image\_list

def save\_images(image\_list):

    index = 0

    for image in image\_list:

        plt.title(index)

        plt.imshow(image)

        plt.show()

        image.save('/content/subimages/'+str(index) + '.png', 'PNG')

        index += 1

def create\_histogram(r,g,b,path):

  ar=np.array(r).flatten()

  plt.hist(ar, bins=256, facecolor='r',edgecolor='r')

  ag=np.array(g).flatten()

  plt.hist(ag, bins=256,  facecolor='g',edgecolor='g')

  ab=np.array(b).flatten()

  plt.hist(ab, bins=256, facecolor='b',edgecolor='b')

  plt.savefig(path)

  plt.show()

def fitter(arr):

  arr=np.array(arr).flatten()

  new\_arr=np.array(arr)

  m=np.where(arr>=max(arr)\*0.8)

  n=np.where(arr<=max(arr)\*0.2)

  for i in m:

    new\_arr[i]=max(arr)\*0.8

  for i in n:

    new\_arr[i]=0

  return new\_arr

def p(arr):

  counter=Counter(arr)#统计各个像素点的个数

  p=np.zeros(256)#计算概率密度

  for i in counter.keys():

    p[i]=counter[i]/sum(counter.values())

  return p

def t(p):

  t=np.zeros(256)

  for i in range(0,256):

    for j in range(0,i):

      t[i]=p[j]+t[i]

  for i in range(0,256):

    t[i]=t[i]\*255+0.5

  return t

def recreate(arr,t):

  shape=np.array(arr).shape

  arr=np.array(arr).flatten()

  new\_img=np.zeros(len(arr))

  for i in range(0,len(arr)):

    new\_img[i]=int(t[arr[i]])

  return im.fromarray(new\_img.reshape(shape)).convert('L')

def getk(path,n):

  img=im.open(path)

  r,g,b,a=img.split()

  r,g,b=np.array(r).flatten(),np.array(g).flatten(),np.array(b).flatten()

  return r[n],g[n],b[n]

def getT(path1,path2,n):#path1 near\_image  path2 aim\_image n index

  img=im.open(path1)

  r,g,b,a=img.split()

  fr,fg,fb=fitter(r),fitter(g),fitter(b)

  Pr,Pg,Pb=p(fr),p(fg),p(fb)

  Tr,Tg,Tb=t(Pr),t(Pg),t(Pb)

  k=getk(path2,n)

  return Tr[k[0]],Tg[k[1]],Tb[k[2]]

def getab(path,n):

  img=im.open(path)

  lenght=len(np.array(img))#32

  a=np.zeros(lenght\*lenght)

  b=np.zeros(lenght\*lenght)

  for i in range(0,lenght):

      for j in range(0,lenght):

        index=n\*i+j

        a[index]=(i-lenght/2)/lenght

        b[index]=(j-lenght/2)/lenght

  return a,b

def create(sub,n):

  path1="/content/subimages/{}.png".format(sub)

  path2="/content/subimages/{}.png".format(sub+1)

  path3="/content/subimages/{}.png".format(sub+n)

  path4="/content/subimages/{}.png".format(sub+1+n)

  a,b=getab(path1,n)

  lenght=len(np.array(im.open(path1)))

  newr=np.zeros(lenght\*lenght)

  newg=np.zeros(lenght\*lenght)

  newb=np.zeros(lenght\*lenght)

  for i in range(0,lenght):

    for j in range(0,lenght):

      index=lenght\*i+j

      print(index,"pixel under processing")

      t1,t2,t3,t4=list([getT(path1,path1,index)][0]),list([getT(path2,path1,index)][0]),list([getT(path3,path1,index)][0]),list([getT(path4,path1,index)][0])

      newr[index]=(1-a[index])\*((1-b[index])\*t1[0]+b[index]\*t2[0])+a[index]\*((1-b[index])\*t3[0]+b[index]\*t4[0])

      newg[index]=(1-a[index])\*((1-b[index])\*t1[1]+b[index]\*t2[1])+a[index]\*((1-b[index])\*t3[1]+b[index]\*t4[1])

      newb[index]=(1-a[index])\*((1-b[index])\*t1[2]+b[index]\*t2[2])+a[index]\*((1-b[index])\*t3[2]+b[index]\*t4[2])

  newr=im.fromarray(newr.reshape((lenght,lenght))).convert('L')

  newg=im.fromarray(newg.reshape((lenght,lenght))).convert('L')

  newb=im.fromarray(newb.reshape((lenght,lenght))).convert('L')

  new\_img=im.merge("RGB",[newr,newg,newb])

  plt.imshow(im.open(path1))

  plt.show()

  plt.imshow(new\_img)

  plt.show()

  new\_img.save(path1)

def recompose(ori\_path,sub\_path,n,m):

   result=im.new("RGB",(128,128))

   img\_list=[]

   for i in range(0,len(os.listdir(sub\_path))):

     img\_list.append("{}/{}".format(sub\_path,i))

   for i in range(0,n):

     for j in range(0,m):

       img=img\_list[i\*n+j]

       result.paste(im.open(img+".png"),box=(j\*int(128/n),i\*int(128/m)))

   plt.imshow(result)

   plt.show()

   result.save("img\_recompose.png")

def one(path,n,m):

  image=(im.open(path)).resize((128,128))

  image\_list = cut\_image(image,n,m)

  save\_images(image\_list)

def two(path):

  img=im.open(path)

  r,g,b,a=img.split()#"L"

  plt.title('oir\_histogram',fontsize='large')

  create\_histogram(r,g,b,"oir\_histogram.png")

  fr,fg,fb=fitter(r),fitter(g),fitter(b)

  plt.title('new\_histogram',fontsize='large')

  create\_histogram(fr,fg,fb,"new\_histogram.png")

  Pr,Pg,Pb=p(fr),p(fg),p(fb)

  Tr,Tg,Tb=t(Pr),t(Pg),t(Pb)

  newr,newg,newb=recreate(r,Tr),recreate(g,Tg),recreate(b,Tb)

  new\_img=im.merge("RGB",[newr,newg,newb])

  plt.title('original image',fontsize='large')

  plt.imshow(img)

  plt.show()

  plt.title('strenghted image',fontsize='large')

  plt.imshow(new\_img)

  plt.savefig("strenght\_image.png")

  plt.show()

  plt.title('original r image',fontsize='large')

  plt.imshow(r)

  plt.show()

  plt.title('original g image',fontsize='large')

  plt.imshow(g)

  plt.show()

  plt.title('original b image',fontsize='large')

  plt.imshow(b)

  plt.show()

  plt.title('strenghted r image',fontsize='large')

  plt.imshow(newr)

  plt.show()

  plt.title('strenghted g image',fontsize='large')

  plt.imshow(newg)

  plt.show()

  plt.title('strenghted b image',fontsize='large')

  plt.imshow(newb)

  plt.show()

def third(ori\_path,sub\_path,n,m):

  for i in range(1,n-1):

    for j in range(1,m-1):

         index=i\*n+j

         create(index,n)

  recompose(ori\_path,sub\_path,n,m)

if \_\_name\_\_ == '\_\_main\_\_':

    n=4

    m=4

    ori\_img = "/content/original\_image.PNG"

    file\_path="/content/subimages"

    sub\_img="/content/subimages/14.png"

    one(ori\_img,n,m)

    two(sub\_img)

    third(ori\_img,file\_path,n,m)