2019 年秋季 《图像处理与分析》编程作业 01

问题 1 黑白图像灰度扫描

实现一个函数 s = scanLine4e(f, I, loc), 其中 f 是一个灰度图像，I 是一个整数，loc 是一个字符串。当 loc 为'row'时，I 代表行数。当 loc 为'column'时，I 代表列数。输出 s 是对应的相关行或者列的像素矢量。

调用该函数，提取 cameraman.tif 和 einstein.tif 的中心行和中心列的像素矢量并将扫描结果绘制成图。

解答：

```
#定义函数
def scanLine4e(f ,I , loc):
    if loc=='row':
        return f[I , :]
    if loc=='column':
        return f[: , I]
```

*#**函数说明**：输入为图像矩阵f、行（列）数I、行列说明字符串loc。针对函数元素loc，通过条件语句将'row'和'column'两种情况分开。当loc='row'时，返回图像矩阵f第I行，即f[I , :]；当loc='column'时，返回图像矩阵第I列，即f[: , I]。*

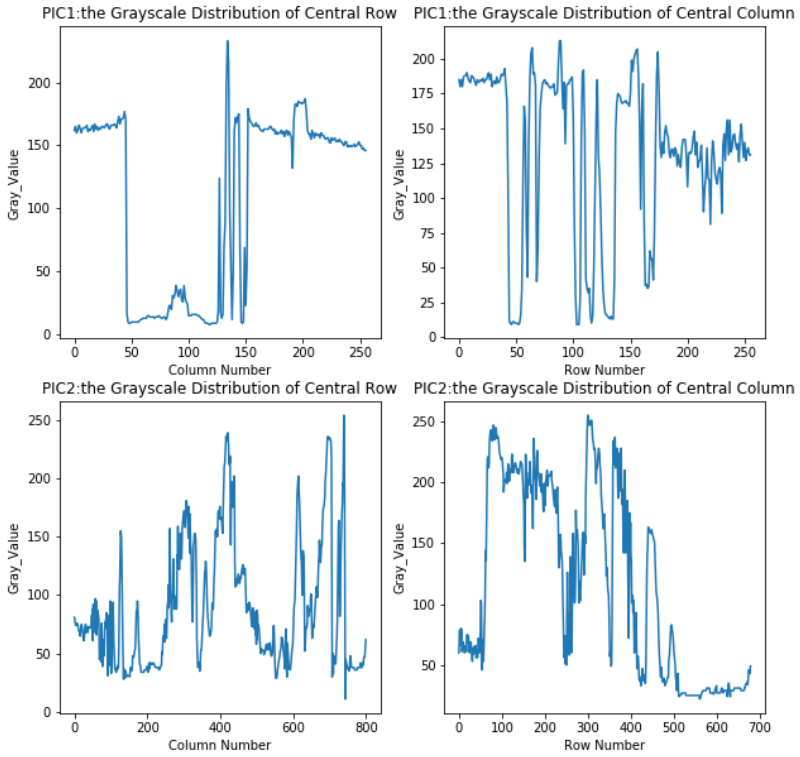
*#**程序运行流程**：1、读取图像矩阵；2、求解矩阵尺寸计算中心行（列）数I值；3、通过函数scanLine4e返回中心行或中心列灰度矢量；4、绘制灰度折线图。*

```
import matplotlib
import numpy as np
import matplotlib.pyplot as plt

plt.figure(figsize=(10,10))
#第一张图片
f1=matplotlib.image.imread('assignment01_images/cameraman.tif')
[m,n]=np.shape(f1) #获取图像矩阵f1 的行数、列数
x1=range(0,n)
x2=range(0,m)
```

```
s1=scanLine4e(f1,m//2,'row') #行数取 I=m//2 即中心行
s2=scanLine4e(f1,n//2,'column') #列数取 I=n//2 即中心列
#绘制中心行灰度分布图
ax1=plt.subplot(221)
ax1.set_title('PIC1:the Grayscale Distribution of Central Row')
ax1.set_xlabel('Column Number')
ax1.set_ylabel('Gray_Value')
plt.plot(x1,s1)
#绘制中心列灰度分布图
ax2=plt.subplot(222)
ax2.set_title('PIC1:the Grayscale Distribution of Central Column')
ax2.set_xlabel('Row Number')
ax2.set_ylabel('Gray_Value')
plt.plot(x2,s2)

#第二张图片
f2=matplotlib.image.imread('assignment01_images/einstein.tif')
[p,q]=np.shape(f2)
x3=range(0,q)
x4=range(0,p)

s3=scanLine4e(f2,p//2,'row')
s4=scanLine4e(f2,q//2,'column')
#绘制中心行灰度分布图
ax3=plt.subplot(223)
ax3.set_title('PIC2:the Grayscale Distribution of Central Row')
ax3.set_xlabel('Column Number')
ax3.set_ylabel('Gray_Value')
plt.plot(x3,s3)
#绘制中心列灰度分布图
ax4=plt.subplot(224)
ax4.set_title('PIC2:the Grayscale Distribution of Central Column')
ax4.set_xlabel('Row Number')
ax4.set_ylabel('Gray_Value')
```

```
plt.plot(x4,s4)

plt.savefig('homework01_1.tif')
```

程序运行结果图：



问题 2　彩色图像转换为黑白图像

图像处理中的一个常见问题是将彩色 RGB 图像转换成单色灰度图像，第一种常用的方法是取三个元素 R，G，B 的均值。第二种常用的方式，又称为 NTSC 标准，考虑了人类的彩色感知体验，对于 R,G,B 三通道分别采用了不同的加权系数，分别是 R 通道 0.2989，G 通道 0.5870，B 通道 0.1140. 实现一个函数 g = rgb1gray(f, method). 函数功能是将一幅 24 位的 RGB 图像, f, 转换成灰度图像, g. 参数 method 是一个字符串，当其值为'average' 时，采用第一种转换方法，当其值为' NTSC'时，采用第二种转换方法。将'NTSC'做为缺省方式。

调用该函数，将提供的图像 mandril_color.tif 和 lena512color.tiff 用上述两种方法转换成单色灰度图像，对于两种方法的结果进行简短比较和讨论。

解答：

```
#定义函数
def rgb1gray(f , method='NTSC'):
    if method=='average':
```

```
            g=f [: , : ,0]/3+f [: , : , 1]/3+f [: , : , 2]/3
            return np.array(g , dtype='uint8')
        if method=='NTSC':
            R=0.2989
            G=0.5870
            B=0.1140
            g=(R*f[: ,: ,0]+G*f[: ,: ,1]+B*f[: ,: ,2])
            return np.array(g,dtype='uint8')
```

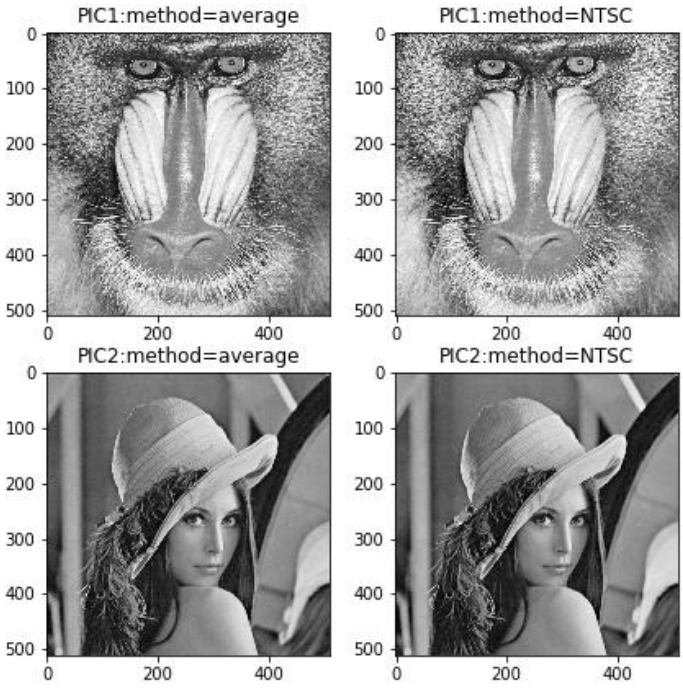
*#**函数说明**：输入为图像矩阵 f、转换方式 method（默认方式为'NTSC'）。当转换方式为'average'时，将三通道矩阵灰度值分别除以 3 相加得到矩阵 g，返回 uint8 型二维图像矩阵；当转换方式为'NTSC'时，RGB 三通道灰度分别乘以比例系数后相加得到矩阵 g，返回 uint8 型二维图像矩阵。当输入中 method 值缺省时，默认使用 NTSC 方式进行转换。*

*#**程序运行流程**：1、读取真彩图像三通道矩阵；2、调用函数，判断转换方式为'average'或'NTSC'。若为前者，将 RGB 三通道矩阵灰度值除以三后相加，返回 uint8 型二维灰度图像矩阵。若为后者，将 RGB 三通道矩阵灰度值按比例相加，返回 uint8 型二维灰度图像矩阵；3、将得到的矩阵绘制成图像并显示。*

```
import matplotlib.pyplot as plt
import numpy as np
#读取图片，调用函数
f1=plt.imread('assignment01_images/mandril_color.tif')
g11=rgb1gray(f1,'average')
g12=rgb1gray(f1,'NTSC')
f2=plt.imread('assignment01_images/lena512color.tiff')
g21=rgb1gray(f2,'average')
g22=rgb1gray(f2,'NTSC')

plt.figure(figsize=(7,7))
#分别绘制图像
ax1=plt.subplot(221)
ax1.set_title('PIC1:method=average')
plt.imshow(g11,cmap="gray")
```

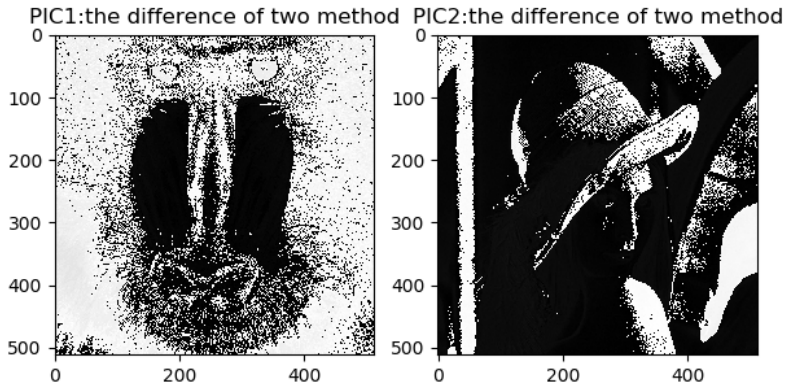
```
ax2=plt.subplot(222)
ax2.set_title('PIC1:method=NTSC')
plt.imshow(g12,cmap="gray")

ax3=plt.subplot(223)
ax3.set_title('PIC2:method=average')
plt.imshow(g21,cmap="gray")

ax4=plt.subplot(224)
ax4.set_title('PIC2:method=NTSC')
plt.imshow(g22,cmap="gray")

 plt.savefig('homework01_2.tif')
```

程序运行结果图：



两种方式的比较：

上述图片经过平均值法以及 NTSC 法进行转换得到的黑白图像从肉眼观察差别较小。但是 NTSC 转换法考虑了人眼对于不同色彩的感知度不同，因此在转换上考虑了不同通道的权重，图像对比度要更好，某些边缘轮廓相较而言会清晰，效果理论上会优于平均值法。为了将差别显示出来，在上述程序后加上两种图片的“减法”运算，得到如下图：



根据图片“减法”运算结果可以看出，两种算法得到的黑白图片是存在区别的，而且在图像中的边缘附近的差别比较明显，能够看见较为明显的轮廓。