一、名词解释

1、数字图像

数字图像是指由被称作像素的小块区域组成的二维矩阵。将物理图像行列划分后，每个小块区域称为像素。
每个像素包括两个属性：位置和亮度（或色彩）

2、图像的灰度直方图

灰度直方图（histogram）是灰度级的函数，描述的是图像中每种灰度级像素的个数，反映图像中每种灰度出现的频率。横坐标是灰度级，纵坐标是灰度级出现的频率。

3、欧氏距离、街区距离和棋盘距离

给定三个像素 p，q，r，其坐标分别为(x,y)，(s,t)，(u,v)，

P 和 q 之间的欧氏距离定义为：

$$D_e(p,q)=[(x-s)^2+(y-t)^2]^{1/2}$$

P 和 q 之间的 D_4 距离（也叫城市街区距离）定义为：

$$D_4(p,q)=|x-s|+|y-t|$$

P 和 q 之间的 D_8 距离（也叫棋盘距离）定义为：

$$D_8(p,q)=\max(|x-s|,|y-t|)$$

4、信息量与信息熵

信息量是随机事件 X 中某件事发生的概率的负对数，它是对信息多少的度量。一句话中的事发生的概率越大，则这句话的信息量越小（“明天太阳会升起”这句话的信息量就很小）。

信息量的表示:

$$h(x) = -log_2 p(x)$$

信息熵

信息量度量的是一个具体事件发生所带来的信息，而熵则是在结果出来之前对可能产生的信息量的期望——考虑该随机变量的所有可能取值，即所有可能发生事件所带来的信息量的期望。即

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log p(x_i)$$

二、简答与证明题

1、证明: 2 维离散灰度图像的均值与其频谱的直流成分成线性关系。

傅里叶变换 $\mathbf{f(x, y)}$的直流成分为 F(0,0)，傅里叶变换公式如下：

$$F(u,v) = \frac{1}{N^2}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)\exp(-j2\pi(ux+vy)/N), \text{v, u=0, 1, 2, 3, 4, 5....N-1}$$

带入得 $F(0,0) = \frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)$

恰好为 二维离散灰度图像的均值

2、高斯拉普拉斯（LoG）算子是经典的边缘检测算子，叙述 LoG 算子的实现过程并推导 LoG 算子的形式。【可能不考】

实现过程：

1、LoG算子

Laplace算子是一种优秀的边缘检测算子，通过对图像求二阶导数，然后通过二阶导数的0交叉点来实现边缘检测。因为Laplace算子对噪声敏感，故可在进行Laplace计算之前用高斯滤波来进行降噪处理，这样就形成了拉普拉斯高斯算子LoG(Laplace of Gaussian)

LOG算子如下：

$$\nabla^2 G = \frac{\partial^2 G}{\partial x^2} + \frac{\partial^2 G}{\partial y^2} = \frac{-2\sigma^2 + x^2 + y^2}{2\pi\sigma^6} e^{-(x^2+y^2)/2\sigma^2}$$

$$G(x,y,\sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$$

推导：

$G(x,y,\sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$ 和 $\mathbf{L(x,y)} = \frac{\partial^2 \mathbf{G}}{\partial \mathbf{x^2}} + \frac{\partial^2 \mathbf{G}}{\partial \mathbf{y^2}}$，这里推导一下。纯手动敲打，有错的话留言提醒一下。

$$\frac{\partial G}{\partial x} = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \left(-\frac{x}{\sigma^2}\right) = -\frac{x}{2\pi\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

$$\frac{\partial^2 G}{\partial x^2} = -\frac{1}{2\pi\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}} + \frac{x^2}{2\pi\sigma^6} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

$$\frac{\partial^2 G}{\partial y^2} = -\frac{1}{2\pi\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}} + \frac{y^2}{2\pi\sigma^6} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

$$\therefore \mathbf{L(x,y)} = \frac{\partial^2 \mathbf{G}}{\partial \mathbf{x^2}} + \frac{\partial^2 \mathbf{G}}{\partial \mathbf{y^2}} = -\frac{1}{\pi\sigma^4}\left[1 - \frac{x^2+y^2}{2\sigma^2}\right] e^{-\frac{x^2+y^2}{2\sigma^2}}$$

3、给出分段线性变换的定义并讨论线性变换的斜率参数取值对输出图像对比度的影响。

定义：概念:有时为了更好地调节图象的对比度，需要在一些亮度段拉伸，而在另一些亮度段压缩，这种变换称为分段线性变换。

分段线性变换：将感兴趣的灰度范围线性扩展，相对抑制不感兴趣的灰度区域。
设 a 为线性变换的斜率：
（1）如果 a>1，输出图像的对比度增大（灰度扩展）
（2）如果 0<a<1,输出图像的对比度减小（灰度压缩）
（3）如果 a 为负值，暗区域将变亮，亮区域将变暗。

4、梯度是灰度图像边缘检测的重要工具，将梯度推广到向量梯度并构造彩色图像边缘检测方法。【可能不考】

灰度图的边缘检测：

回顾灰度图像边缘检测，我们一般求出图像的梯度:

$$\nabla f_x(x,y) = f(x+1,y) - f(x,y)$$
$$\nabla f_y(x,y) = f(x,y+1) - f(x,y)$$

或者求出图像的二阶差分:

$$\Delta f_x(x,y) = f(x+1,y) + f(x-1,y) - 2f(x,y)$$
$$\Delta f_y(x,y) = f(x,y+1) + f(x,y-1) - 2f(x,y)$$

然后根据梯度和二阶差分的大小,以及最大变化率方向来寻找可能存在的图像边缘。

彩图的边缘检测：

彩色图像的每个像素包含红绿蓝三个分量，这样每个像素可以由一个三维向量来表示。但是在进行图像边缘检测的时候，我们遇到一个问题，那就是向量并不存在梯度概念。单独对每个颜色分量进行边缘检测，其梯度不能反映图像整体彩色的差异变化。

一个广为使用的彩色图像梯度方法来自Zenzo[1986]的论文

$$\vec{u} = \frac{\partial R}{\partial x}\vec{r} + \frac{\partial G}{\partial x}\vec{g} + \frac{\partial B}{\partial x}\vec{b}$$
$$\vec{v} = \frac{\partial R}{\partial y}\vec{r} + \frac{\partial G}{\partial y}\vec{g} + \frac{\partial B}{\partial y}\vec{b}$$

其中R,G,B是图像分量，$\vec{r},\vec{g},\vec{b}$等是单位向量，表征颜色分量坐标。然后继续计算有:

$$g_{xx} = \vec{u}^T\vec{u} = \left|\frac{\partial R}{\partial x}\right|^2 + \left|\frac{\partial G}{\partial x}\right|^2 + \left|\frac{\partial B}{\partial x}\right|^2$$
$$g_{yy} = \vec{v}^T\vec{v} = \left|\frac{\partial R}{\partial y}\right|^2 + \left|\frac{\partial G}{\partial y}\right|^2 + \left|\frac{\partial B}{\partial y}\right|^2$$
$$g_{xy} = \vec{u}^T\vec{v} = \frac{\partial R}{\partial x}\frac{\partial R}{\partial y} + \frac{\partial G}{\partial x}\frac{\partial G}{\partial y} + \frac{\partial B}{\partial x}\frac{\partial B}{\partial y}$$

注意这里执行的是向量乘法，而不是进一步求导。梯度方向为:

$$\theta = \frac{1}{2}\arctan\left[\frac{2g_{xy}}{(g_{xx} - g_{yy})}\right]$$

由梯度方向才能计算梯度:

$$F(\theta) = \{\frac{1}{2}[(g_{xx} + g_{yy}) + (g_{xx} - g_{yy})\cos 2\theta + 2g_{xy}\sin 2\theta]\}^{\frac{1}{2}}$$

matlab上自带的edge函数目前没发现可以对彩色图像直接进行边缘检测的方法，C++的openCV库也没有发现。上述工具一般会对彩色图像进行灰度化再执行边缘检测，效果也不算差。

Zenzo, A Note on the Gradient of a Multi-Image,° Computer[C]// Vision Graphics Image Processing. 1986.

三、计算题

1、给出 Prewritt 算子对应模板的公式，对如下图所示数据计算梯度向量的值（忽略最外侧的数据），并近似计算梯度的模。

218	213	207	189	173
217	212	199	179	163
211	201	189	171	157
207	196	181	166	153
200	180	168	157	144
198	175	164	154	148
189	170	159	150	145

X 方向

-1		1
-1		1
-1		1

Y 方向

1	1	1
-1	-1	-1

计算过程如下（以第一个点为例）：

-1* （218+217+211）+1* （207+199+189）=-51

最终全部计算完会得到矩阵：

-51	-87	-102	37	48	52
-66	-93	-96	44	47	41
-80	-83	-84	53	56	48
-92	-74	-68	47	50	34
-96	-64	-54	30	26	15

梯度的模(题目中要求**近似**，故用绝对值相加，否则用平方方加和再开方)：

M=｜tx｜+｜ty｜

得矩阵：

88	135	154
110	140	137
133	139	132
139	124	102
126	90	69

2、给出欧拉数的计算公式，并计算 2018 四个字母的欧拉数。

（对于此题，我们暂不考虑联通类型）

对于二维图像，欧拉数 E = C - H，其中 C 为连接体数，H 为孔洞数

对于数字 2：连接体数为 1，孔洞数为 0，欧拉数为 1

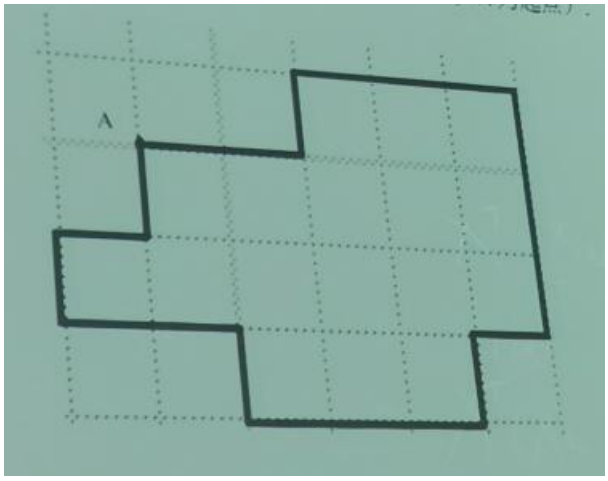
对于数字 0：连接体数为 1，孔洞数为 1，欧拉数为 0

对于数字 1：连接体数为 1，孔洞数为 0，欧拉数为 1

对于数字 8：连接体数为 1，孔洞数为 2，欧拉数为-1

故 2018 四个字母的欧拉数分别为 1，0，1，-1。

3、计算如下图所示区域边界的 4 向链码、归一化链码和差分链码(以 A 为起点)。



四向链码：0010003332322212210 1
归一化链码：0003332322212210 1001
差分链码：3013003003130031033 1

4、给出从 RGB 彩色模型到 HIS 彩色模型的转换公式，并计算 RGB 颜色空间中（225，240，0）在 HIS 颜色模型中的取值（如需要，可以用开方、反余弦等函数来表示）。

RGB转HSI公式：

$$I = \frac{1}{3}(R+G+B)$$

$$S = 1 - \frac{3}{(R+G+B)}[min(R,G,B)]$$

$$H = \arccos \frac{[(R-G)+(R-B)]/2}{[(R-G)^2+(R-B)^2+(G-B)^2]^{0.5}}$$

将（255,240,0）归一化：

$$r = R/(R+G+B);\ g = G/(R+G+B);\ b = B/(R+G+B)$$

得（0.515，0.485，0）
带入得HSI取值为（0.333，1，arccos(0.273/√1.889)）

四、算法编程

边缘检测是图像处理中的重要内容，Photoshop 软件中的查找边缘滤镜是利用边缘检测技术实现的，其实现过程是对输入的一幅 RGB 彩色图像，对每个通道使用 Prewitt 算子计算梯度的模，并将其合并为彩色图像，最后进行反相处理，按照上述流程设计算法流程并进行编程实现，要求对关键步骤添加简要注释。

算法编程-Python：

```
import cv2
import numpy as np

img = cv2.imread('img.png') # 以 BGR 读入图片

cv2.imshow('src',img)
img = img.astype('float64')

kenelx = np.array([[-1,0,1],[-1,0,1],[-1,0,1]])  # 垂直算子
kenely = np.array([[-1,-1,-1],[0,0,0],[1,1,1]])  # 水平算子

prewittx = cv2.filter2D(img,-1,kenelx)  # 二维卷积，计算垂直梯度
prewitty = cv2.filter2D(img,-1,kenely)  # 二维卷积，计算水平梯度
prewittimg = np.sqrt(np.square(prewittx) + np.square(prewitty))  # 计算梯度的模，绝对值加和是梯度模的近似

# 计算模后 img 的像素值分布不再是 0-255，而可能超过了 255，为了回到 0-255，需要归一化
max, min = np.max(prewittimg), np.min(prewittimg)
newimg = (prewittimg-min)/(max-min)*255
newimg = 255 - newimg  # 反色操作
newimg = newimg.astype('uint8')

cv2.imshow('new_img',newimg)
cv2.waitKey(0)
```

算法编程-Matlab：

```
clc
clear
close all

f = imread('qie.jpg');
R = f(:,:,1);
G = f(:,:,2);
B = f(:,:,3);
```

```
[Rx,Ry] = forprewitt(R);
[Gx,Gy] = forprewitt(G);
[Bx,By] = forprewitt(B);
imshow(Rx,[]),title('红图：X');
figure,imshow(Ry,[]),title('红图：Y');
i = sqrt(Rx.^2+Ry.^2);
figure,imshow(i,[]),title('红图：取模');
figure,imshow(Gx,[]),title('绿图：X');
figure,imshow(Gy,[]),title('绿图：Y');
i = sqrt(Gx.^2+Gy.^2);
figure,imshow(i,[]),title('绿图：取模');
figure,imshow(Bx,[]),title('蓝图：X');
figure,imshow(By,[]),title('蓝图：Y');
i = sqrt(Bx.^2+By.^2);
figure,imshow(i,[]),title('蓝图：取模');
RGBnorm(:,:,1) = R;
RGBnorm(:,:,2) = G;
RGBnorm(:,:,3) = B;
figure,imshow(RGBnorm,[]),title('彩图');
RGBnorm(:,:,1) = sqrt(Rx.^2+Ry.^2);
RGBnorm(:,:,2) = sqrt(Gx.^2+Gy.^2);
RGBnorm(:,:,3) = sqrt(Bx.^2+By.^2);
figure,imshow(RGBnorm,[]),title('彩图:取模');
I_inverse =  imcomplement(RGBnorm);
figure,imshow(I_inverse,[]),title('彩色图像边缘检测反相后')
```

平时实验

彩色图像变灰度图

```
from PIL import Image
import numpy as np
import matplotlib.pyplot as plt

# 读入图像
img = Image.open('test.jpg')
img = np.array(img)
if img.ndim == 3:
    img = img[:,:,0]

plt.imshow(img, cmap = plt.cm.gray)
```

分段线性变换、对数变换、指数变换、通道混合器

```
import matplotlib.pyplot as plt
import math
import numpy as np
from PIL import Image
'''''读入图像'''
path='test.jpg'
img=Image.open(path).convert('L')
plt.figure(figsize=(20, 8))
plt.imshow(img,'gray')

'''''分段线性变换'''
img_array=np.array(img)
for i in range(0,img.size[1]-1):
    for j in range(0,img.size[0]-1):
        if img_array[i][j]<=30:
            img_array[i][j]=img_array[i][j]
        elif img_array[i][j]<=150:
            img_array[i][j]=(200-30)/(150-30)*(img_array[i][j]-30)+30
        else :
            img_array[i][j]=(255-200)/(255-150)*(img_array[i][j]-150)+200
plt.figure(figsize=(20, 8))
plt.imshow(img_array,'gray')

'''''幂律变换'''
def fun_log(x):
    return math.log(1+x)*41
img_log=Image.eval(img,fun_log)
plt.figure(figsize=(20, 8))
plt.imshow(img_log,'gray')

'''''指数变换'''
def fun_pow(x):
    return math.pow(x,0.4)
img_pow=Image.eval(img,fun_pow)
plt.figure(figsize=(20, 8))
plt.imshow(img_pow,'gray')

'''''通道混合器'''
img_2=Image.open(path)
pix=np.array(img_2)
R=pix[:,:,0]
```

```
G=pix[:,:,1]
B=pix[:,:,2]
w1,w2,w3=0.1,0.2,0.35
Rw=w1*R+w2*G+w3*B
pix[:,:,0]=R
pix[:,:,1]=G
pix[:,:,2]=Rw
plt.figure(figsize=(20, 8))
plt.imshow(pix,'gray')
```

cDNA

```
import matplotlib.pyplot as plt
import cv2
import numpy as np
from PIL import Image

def img_line_split(img):
    a = img.sum(axis = 0)
    b = img.sum(axis = 1)
    key_point_row = []
    key_point_col = []
    MAXX = 255*img.shape[0]
    l = 0
    r = 0
    for i in range(1,a.shape[0]-1):
        if a[i+1]!=0 and a[i]==0 and i!=820:
            r = i
            key_point_row.append(int((l+r)/2))
        elif a[i]!=0 and a[i+1]==0 and i!= 820:
            l = i+1
    l = 0
    r = 0
    for i in range(1,b.shape[0]-1):
        if b[i+1]!=0 and b[i]==0 and i!=820:
            r = i
            key_point_col.append(int((l+r)/2))
        elif b[i]!=0 and b[i+1]==0 and i!= 820:
            l = i+1
    #print(key_point_row)
    #print(key_point_col)
    for i in range(img.shape[0]):
        for j in key_point_col:
```

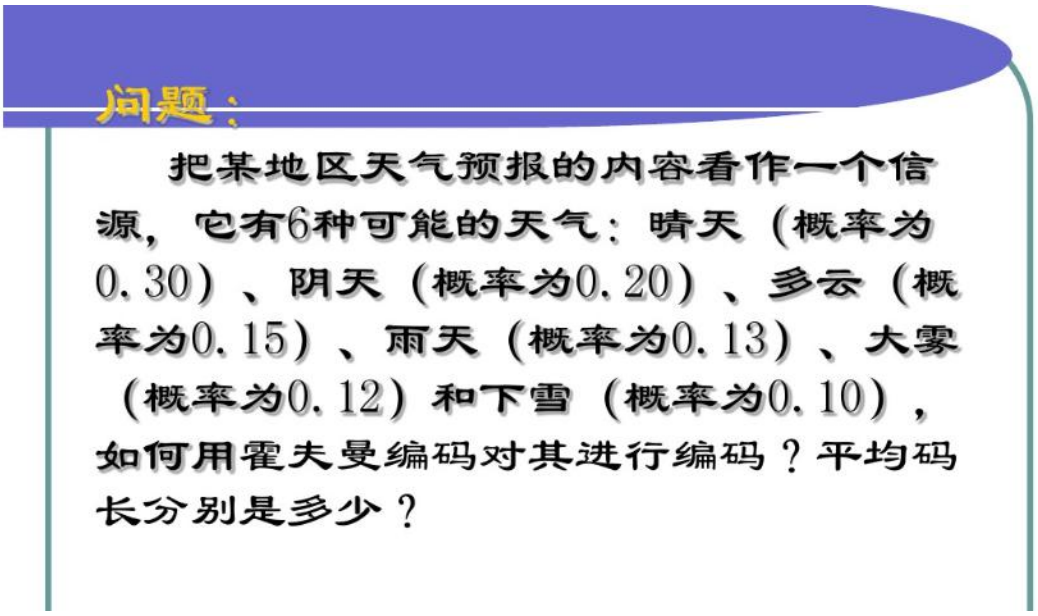
```
            img[i,j] = 255
    for i in key_point_col:
        for j in range(img.shape[0]):
            img[i,j] = 255
    return img
def img_convert(img):
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            if img[i,j] >100:
                img[i,j] = 255
            else:
                img[i,j] = 0
if __name__ =="__main__":
    #路径
    path = r'C:\Users\Ning Hui\Desktop\cDNA.png'
    #读取图像
    img = cv2.imread(path)
    #转化为灰度图像
    grey_img = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    img_split= img_split(grey_img)
    final_img = img_convert(img_split)
    #grey_img = col_sum(row_img)
    cv2.imshow('final_img',img_split)
    cv2.waitKey(0)
    #分割三通道
    img1 = cv2.imread(path)
    b,g,r = cv2.split(img1)
    ls = [b,g,r]
    for i in range(3):
        ls[i]= img_split(ls[i])
        ls[i]= img_convert(ls[i])
    img1[0,:,:],img1[1,:,:],img1[2,:,:] = ls[0],ls[1],ls[2]

    cv2.imshow('final_img',img1)
    cv2.waitKey(0)

```

读取图像获取信息，八邻域

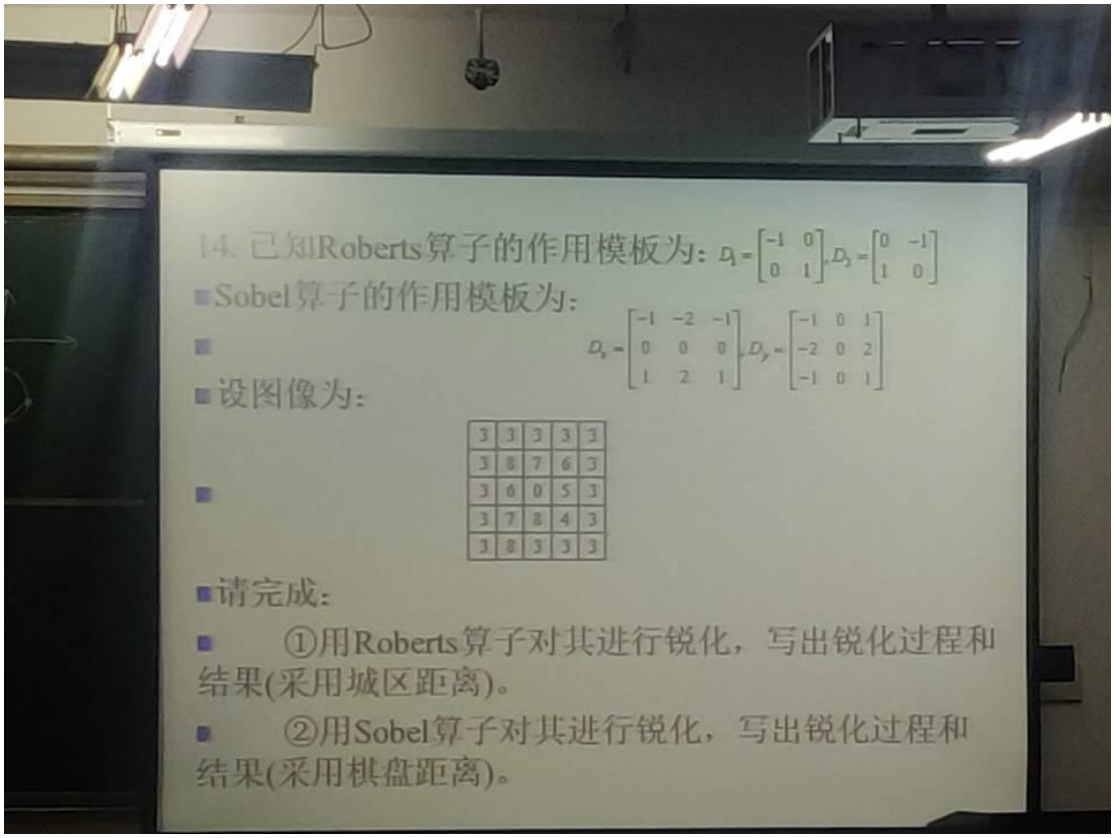
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
path_1 = r'C:\Users\Ning Hui\Desktop\1.jpg'
```

```
path_2 = r'C:\Users\Ning Hui\Desktop\2.jpg'
#读取一张彩色图并转化为灰度图像
img_grey2 = cv2.imread(path_1,cv2.IMREAD_GRAYSCALE)

cv2.imshow('gray image',img_grey2)
cv2.waitKey(0)

#311809000521   19990408
m,n = 21,107
#打印(m,n)所在的像素值

print(f'坐标({m},{n})的像素值为{img_grey2[m][n]}')

#获取(m,n)像素值的八邻域的像素值，返回一个 3*3 矩阵
partial = img_grey2[m-1:m+2,n-1:n+2]
#显示像素值
print(f'坐标({m},{n})的 8 邻域像素值为{partial}')
#为了计算方便，将(m,n)像素值设置为 0
partial[1][1] = 0
#求矩阵所有值之和，并求平均值
mean_value = np.sum(partial)/8
#打印均值
print(f'8 邻域的均值为{mean_value}')
#获取最大值和最小值的坐标
mx = np.unravel_index(np.argmax(partial),partial.shape)
mi = np.unravel_index(np.argmin(partial),partial.shape)
dis_max_min = np.sqrt((mx[1]-mi[1])**2+(mx[0]-mi[0])**2)
#打印距离
print(f'最大值点最小值点的欧式距离为{dis_max_min}')
```

平时作业



解题参考：http://www.doc88.com/p-7754387411649.html



Prewitt 算子和 Sobal 算子

参考

【图像处理】彩色图像边缘检测：https://blog.csdn.net/lpsl1882/article/details/51802006/
机器视觉——2019 试卷：https://blog.csdn.net/weixin_44049693/article/details/110899323