

作业要求：

读入摄像头，回放视频。按一下空格键，则暂停回放，并将当前帧图像做一次Harris Corner检测，并将检测的结果叠加在原来图像上

1. 需要自己写代码实现Harris Corner检测算法，不能直接调用OpenCV里面与Harris角点检测相关的函数；
2. 显示中间的处理结果及最终的检测结果，包括最大特征值图，最小特征值图，R图(可以考虑彩色展示)，原图上叠加检测结果等，并将这些中间结果都输出保存为图像文件。

信息：

姓名：鲍奕帆

学号：3180103499 邮箱：[3180103499@zju.edu.cn](mailto:3180103499@zju.edu.cn)

完成日期：2020年12月15日晚上10点

## 一、软件开发说明

---

- 计算机与操作系统

macOS Catalina 版本 10.15.5

MacBook Pro(13-inch, 2020, Four Thunderbolt 3 ports)

CPU: 2 GHz 四核 Intel Core i5

内存: 16 GB 3733 MHz LPDDR4

启动磁盘: Macintosh HD

图形卡: Intel Iris Plus Graphics 1536 MB

内核(uname -a): Darwin EvandeMacBook-Pro.local 19.5.0 Darwin Kernel Version 19.5.0: Tue May 26 20:41:44 PDT 2020; root:xnu-6153.121.2~2/RELEASE\_X86\_64 x86\_64

- opencv 版本

opencv\_4.4.0

- 编程语言

python 3.8.5

- 其他的包

numpy

## 二、实验结果展示与分析

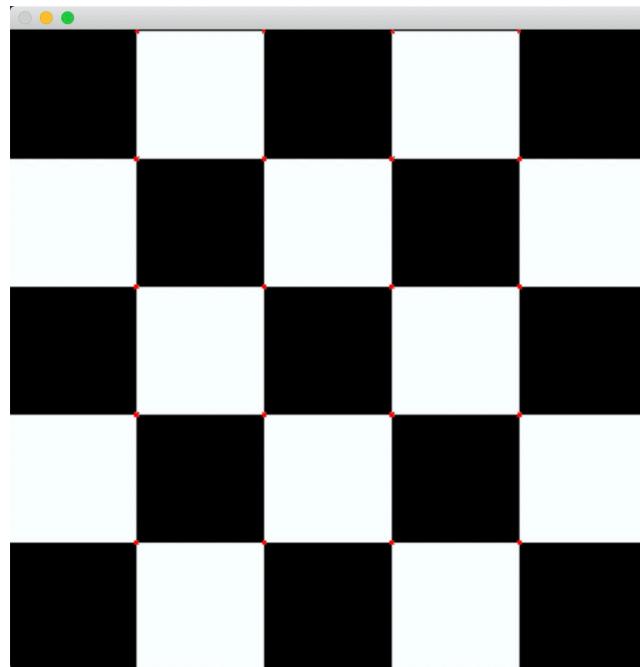
---

1. 角点检测结果(这里展示的包括了静态图片展示，以及动态图片人脸展示)

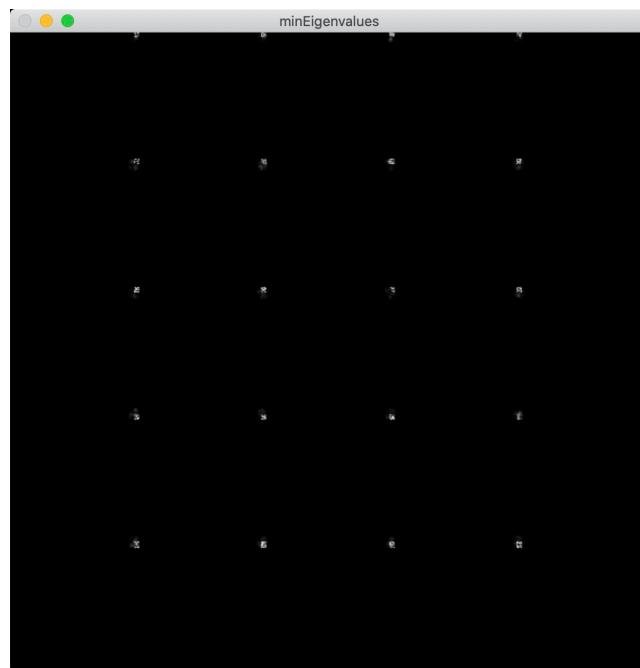
注意：由于采取的库和语言的不同，最终的结果可能有一些出入。这里采用的是python的一些方法，经过检验，最终的角点检测结果与python中的cornerHarris类似，效果较好。但中间结果，如最大特征值图、最小特征值图以及R值图可能数据范围不同，导致最终的显示效果未必很好。

- 检测图1--方格图

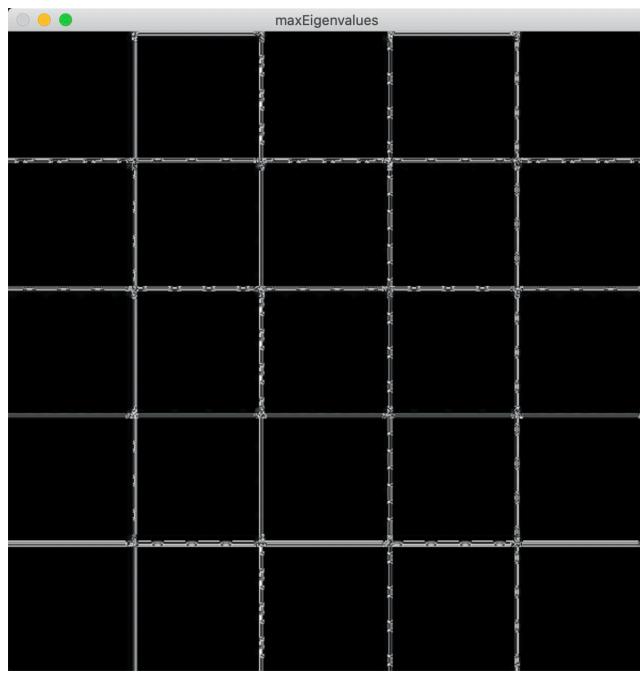
原图叠加检测点（红色为corner点）



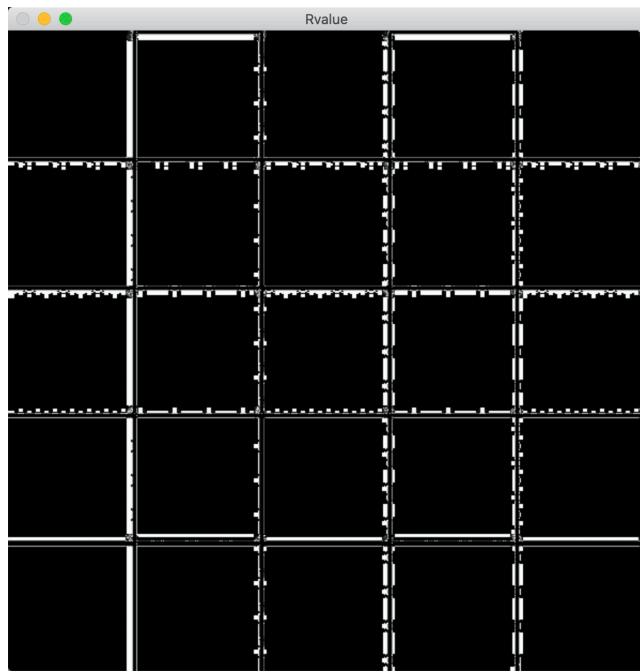
最小特征值图



最大特征值图



R图



- 检测图2 长劲鹿检测

原图叠加检测点



最小特征值图



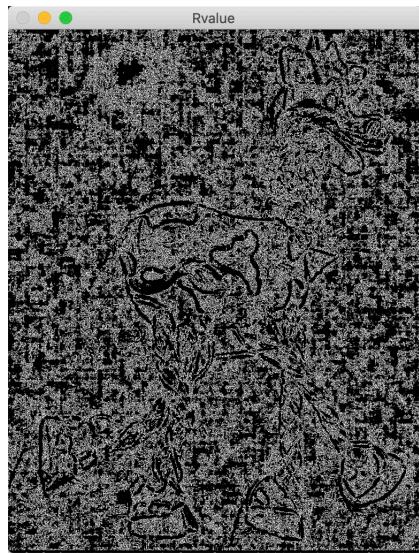
最大特征值图



R图(选择了一定的threshold)



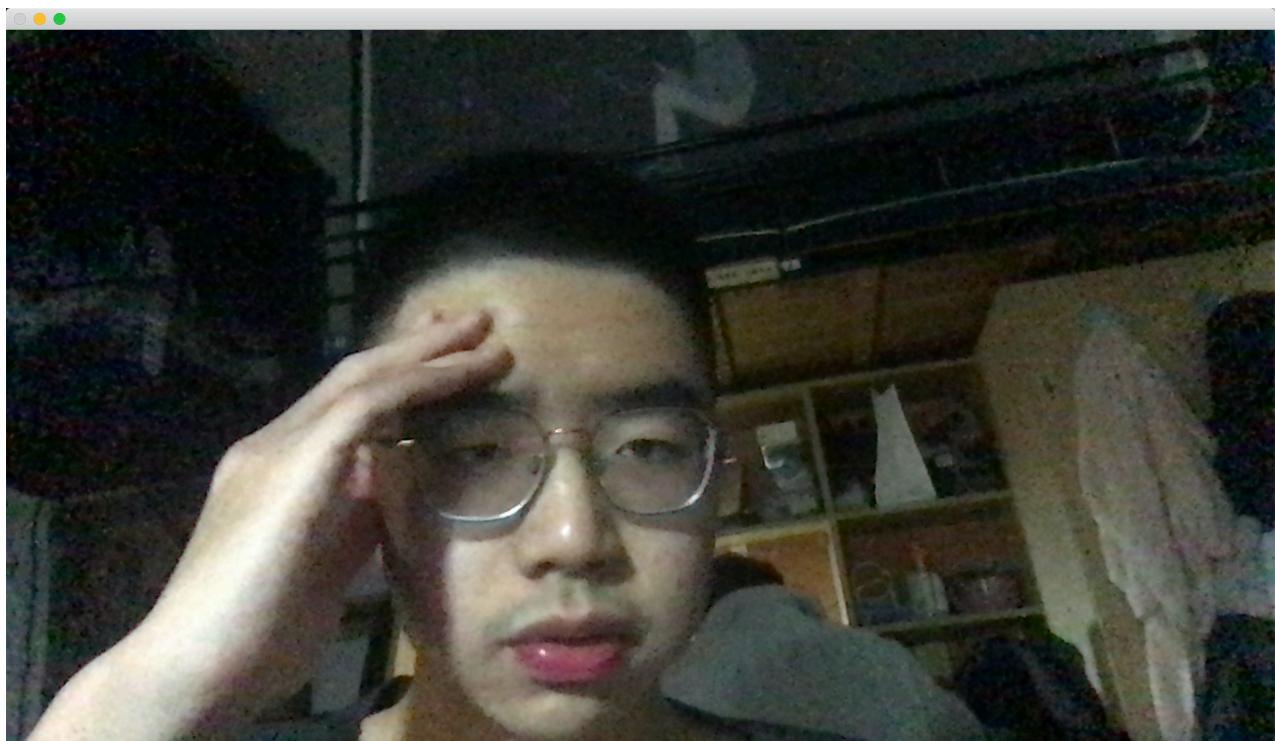
R图(无threshold)



这里主要还是python里面各种数据结构导致的结果不清晰。因为有各种数据类型，浮点、符号整数、无符号整数，以及不同的位数。cv2.imshow的显示也有自己的要求。这就导致最终显示的时候出现各种数据的化简与截断。但并不影响最终结果的正确性。

- 视频拍摄检测

人脸corner检测效果。总体来看还是很不错的。和python的opencv对比了一下，结果比较类似，但自己实现的多了一些不必要的点。但系统中的函数给出的角点更多，且运算速度更快。自己实现的内容，由于是python代码，运算速度极慢。好处是易于编写。



仔细观察！！手臂上的红点，以及眼镜框上的红点。其实是很明显的，就是光线比较暗，且与红色差异不明显。

最小特征值图(后面重拍)



最大特征值图



### 三、算法描述

- Harris Corner 算法简要说明

1. 定义灰度变化描述

$$E(u, v) = \sum w(x, y) [I(x+u, y+v) - I(x, y)]^2$$

- 这里 $u, v$ 是窗口W的偏移量
- $(x,y)$ 是窗口W中像素点的坐标
- $I(x,y)$ 是 $(x,y)$ 坐标像素的灰度值,  $I(x+u,y+u)$ 是像素坐标 $(x+u,y+u)$ 的灰度值
- W是窗口函数。一般为二元正太函数(离散为高斯核)。这是因为如果窗口W中心点是角点时, 移动前与移动后, 该点对灰度变化贡献最大; 而该角点角远点点, 灰度变化几乎平缓, 因此对于灰度变化贡献较小。

## 2. 根据泰勒公式逼近 $E(u,v)$

$$\begin{aligned}
 E(u,v) &= \sum_{x,y} w(x,y) [I(x+u,y+v) - I(x,y)]^2 \\
 &= \sum_{x,y} w(x,y) [I(x,y) + uI_x(x,y) + vI_y(x,y) - I(x,y)]^2 \\
 &= \sum_{x,y} w(x,y) [uI_x(x,y) + vI_y(x,y)]^2 \\
 &= [u, v] \left( \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \right) \begin{bmatrix} u \\ v \end{bmatrix}
 \end{aligned}$$

$$E(u,v) \approx [u, v] M \begin{bmatrix} u \\ v \end{bmatrix}$$

其中M

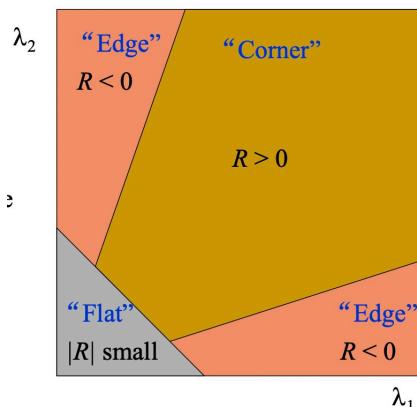
$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

## 3. 定义角点度量函数R(核心! )

$$R = \det M - k (\operatorname{trace} M)^2$$

$$\begin{aligned} \det M &= \lambda_1 \lambda_2 \\ \operatorname{trace} M &= \lambda_1 + \lambda_2 \end{aligned}$$

## 4. 根据R判断角点



基于此得到Harris Corner检测算法基本步骤

a. 计算图像 $I(x,y)$ 在 $x$ 和 $y$ 两个方向的梯度 $I_x, I_y$

对于离散情况，采用差分近似微分。本实验采用sobel算子计算

b. 计算图像两个方向梯度的乘积

c. 使用高斯函数(窗口函数)对 $I_x^2$ 、 $I_y^2$ 、 $I_x I_y$ 进行加权求和，计算中心点为 $(x,y)$ 的窗口对应的矩阵 $M$

$$A = \sum_{(x,y) \in W} g(I_x^2) = \sum_{(x,y) \in W} I_x^2 * w(x, y)$$

$$B = \sum_{(x,y) \in W} g(I_y^2) = \sum_{(x,y) \in W} I_y^2 * w(x, y)$$

$$C = \sum_{(x,y) \in W} g(I_x I_y) = \sum_{(x,y) \in W} I_x I_y * w(x, y)$$

d. 根据前面公式计算图像上所有像素点 $(x,y)$ 对应的Harris响应值R

e. 过滤出大于某个阈值的R值，利用局部极大值寻找角点。

## 四、程序结构描述

- 源文件

由于是python文件，因此仅有一个文件，即corner.py。

内部函数有normalization(data)和HarrisCorner(img, ksize=3)，分别用于实现正则化和Harris Corner检测。

下面简要叙述一下"main"函数的功能，也就是主要实现了读入摄像头，回放视频。按一下空格键，则暂停回放，最后调用自己实现的HarrisCorner函数进行检测与图像输出保存。

main函数代码如下

```
if __name__ == '__main__':
    cap = cv2.VideoCapture(0)
    while(1):
        # get a frame
        ret, frame = cap.read()
        # show a frame
        cv2.imshow("capture", frame)
        if cv2.waitKey(1) & 0xFF == ord(' '):
            cv2.waitKey(0)
            # #转换为灰度图像
            img = frame
            imgShape = img.shape
            gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
            value, dst = HarrisCorner(gray)
            minValue = value[:, 0] #最小特征值
            maxValue = value[:, 1] #最大特征值
            minValue = np.reshape(minValue, (imgShape[0], imgShape[1]))
            maxValue = np.reshape(maxValue, (imgShape[0], imgShape[1]))
```

```

        minValue=minValue.astype(np.uint8)
        maxValue=maxValue.astype(np.uint8)
        cv2.imshow('minEigenvalues',minValue)
        cv2.imwrite('minEigenvalues.jpg',minValue)
        cv2.imshow('maxEigenvalues',maxValue)
        cv2.imwrite('maxEigenvalues.jpg',maxValue)
        frame[dst>0.01*dst.max()] = [0,0,255]
        cv2.imshow('result',frame)
        cv2.imwrite('result.jpg',frame)
        cv2.waitKey(0)
        cv2.destroyAllWindows()

#cv2.waitKey(0)

cap.release()
cv2.destroyAllWindows()

```

事实上就是比较简单的读入输出,以及显示。

基本过程

1. 外层while循环-保证视频的连续显示, cv2.VideoCapture(0)从摄像头获取拍摄权限
2. cap.read()获得读取到的每一帧。
3. cv2.imshow("capture", frame)显示视频内容, 这里采用默认的fps
4. if cv2.waitKey(1) & 0xFF == ord(' ') 用于判断空格, 暂停视频
5. 内部调用HarrisCorner获得corner以及特征值, 将内容写入与输出--R值的写入在HarrisCorner函数中实现
6. 最后释放窗口

## 五、算法具体实现

此处讲解自己实现的HarrisCorner算法的python代码内容; 其基本算法思想和步骤与三、算法描述中的a-e步骤一一对应

代码与注释如下

```

def HarrisCorner(img, ksize=3):
    k = 0.04 # 经验值k-响应函数计算公式中
    threshold = 0.01 # threshold 阈值 用于选择corner
    # 采用NMS非极大值抑制 会导致最终选择到的点过少 -- 这里没有采用
    height, width = img.shape[:2] # 获取图像高和宽
    # 使用sobel算子计算两个方向的梯度
    grad = np.zeros((height,width,2),dtype=np.float32)
    # 注意这里采用cv2.CV_16S主要是因为梯度计算的值有负数
    grad[:, :, 0] = cv2.Sobel(img, cv2.CV_16S, 1, 0, ksize=3) # x方向梯度
    grad[:, :, 1] = cv2.Sobel(img, cv2.CV_16S, 0, 1, ksize=3) # y方向梯度
    # m矩阵相关参数计算//四个角的值
    m = np.zeros((height,width,3),dtype=np.float32)
    m[:, :, 0] = grad[:, :, 0]**2 # Ix^2,

```

```

m[:, :, 1] = grad[:, :, 1]**2 # Iy^2
m[:, :, 2] = grad[:, :, 0]*grad[:, :, 1] # Ix*Iy
# m矩阵计算 利用高斯卷积
m[:, :, 0] = cv2.GaussianBlur(m[:, :, 0], ksize=(ksize, ksize), sigmaX=2)
m[:, :, 1] = cv2.GaussianBlur(m[:, :, 1], ksize=(ksize, ksize), sigmaX=2)
m[:, :, 2] = cv2.GaussianBlur(m[:, :, 2], ksize=(ksize, ksize), sigmaX=2)
#计算得m矩阵
m = [np.array([[m[i, j, 0], m[i, j, 2]], [m[i, j, 2], m[i, j, 1]]]) for i in
range(height) for j in range(width)]
# 根据M矩阵 计算特征值---用于完成中间步骤图的绘制
value, vector = np.linalg.eig(m)
value.sort()
# 计算M矩阵的行列式 与 迹
D, T = list(map(np.linalg.det, m)), list(map(np.trace, m))
# 根据结果计算响应函数R---R(i,j)=det(M)-k(trace(M))^2
R = np.array([d-k*t**2 for d, t in zip(D, T)])
# 阈值获取角点
R_max = np.max(R)
R = R.reshape(height, width)

R_norm = normalization(R)
# R_norm[R_norm<150] = 0
# R_norm[R_norm>=150]=255
cv2.imshow('Rvalue', R_norm.astype(np.uint8))
cv2.imwrite('Rvalue.jpg', R_norm.astype(np.uint8))

corner = np.zeros_like(R, dtype=np.uint8)
for i in range(height):
    for j in range(width):
        if R[i, j] > R_max*threshold :
            corner[i, j]=1 # flag 表明是大于阈值的corner
# 寻找局部极大值 设置角点值为255
for i in range(1, height-1):
    for j in range(1, width-1):
        if corner[i, j]==1 and R[i][j] > R[i-1][j] and R[i][j] > R[i+1][j]
and R[i][j] > R[i][j+1] and R[i][j] > R[i][j-1]:
            corner[i, j]=255 # 满足局部极大值 设置为255
return value, corner

```

针对算法步骤展开

a. 计算图像 $I(x,y)$ 在 $x$ 和 $y$ 两个方向的梯度 $I_x, I_y$

```

grad = np.zeros((height, width, 2), dtype=np.float32)
# 注意这里采用cv2.CV_16S主要是因为梯度计算的值有负数
grad[:, :, 0] = cv2.Sobel(img, cv2.CV_16S, 1, 0, ksize=3) # x方向梯度
grad[:, :, 1] = cv2.Sobel(img, cv2.CV_16S, 0, 1, ksize=3) # y方向梯度

```

直接采用cv2.Sobel计算离散梯度。

对于离散情况，采用差分近似微分。本实验采用sobel算子计算

b. 计算图像两个方向梯度的乘积

```
m[:, :, 0] = grad[:, :, 0]**2 # Ix^2,  
m[:, :, 1] = grad[:, :, 1]**2 # Iy^2  
m[:, :, 2] = grad[:, :, 0]*grad[:, :, 1] # Ix*Iy
```

直接利用运算\*\*进行bitwise乘幂

c. 使用高斯函数(窗口函数)对 $I_x^2$ 、 $I_y^2$ 、 $I_x I_y$ 进行加权求和，计算中心点为(x,y)的窗口对应的矩阵M

$$A = \sum_{(x,y) \in W} g(I_x^2) = \sum_{(x,y) \in W} I_x^2 * w(x,y)$$
$$B = \sum_{(x,y) \in W} g(I_y^2) = \sum_{(x,y) \in W} I_y^2 * w(x,y)$$
$$C = \sum_{(x,y) \in W} g(I_x I_y) = \sum_{(x,y) \in W} I_x I_y * w(x,y)$$

```
m[:, :, 0] = cv2.GaussianBlur(m[:, :, 0], ksize=(ksize, ksize), sigmaX=2)  
m[:, :, 1] = cv2.GaussianBlur(m[:, :, 1], ksize=(ksize, ksize), sigmaX=2)  
m[:, :, 2] = cv2.GaussianBlur(m[:, :, 2], ksize=(ksize, ksize), sigmaX=2)
```

高斯函数加权求和///即用高斯核卷积

d. 根据前面公式计算图像上所有像素点(x,y)对应的Harris响应值R

```
# 计算M矩阵的行列式 与 迹  
D, T = list(map(np.linalg.det, m)), list(map(np.trace, m))  
# 根据结果计算响应函数R---R(i,j)=det(M)-k(trace(M))^2  
R = np.array([d-k*t**2 for d, t in zip(D, T)])
```

R的计算。

e. 过滤出大于某个阈值的R值，利用局部极大值寻找角点。

```
corner = np.zeros_like(R, dtype=np.uint8)  
for i in range(height):  
    for j in range(width):  
        if R[i, j] > R_max*threshold :  
            corner[i, j]=1 # flag 表明是大于阈值的corner  
# 寻找局部极大值 设置角点值为255  
for i in range(1, height-1):  
    for j in range(1, width-1):  
        if corner[i, j]==1 and R[i][j] > R[i-1][j] and R[i][j] > R[i+1][j]  
and R[i][j] > R[i][j+1] and R[i][j] > R[i][j-1]:  
            corner[i, j]=255 # 满足局部极大值 设置为255
```