

**计算机视觉作业报告**

|  |  |
| --- | --- |
| 作业名称： | Harris Corner Detection |
| 姓 名： | 胡单春 |
| 学 号： | 21921082 |
| 电子邮箱： | 3150102279@zju.edu.cn |
| 联系电话： | 15724998468 |
| 导 师： | 邵健 |

2019年 12 月 11 日

**Harris Corner Detection**

1. 作业已实现的功能简述及运行简要说明

**已实现功能：**

1：利用argparse模块实现自定义命令行参数file\_path，以此读取用户希望处理的文件。

2：读取一张彩色图片，完成Harris Corner Detection算法，将最大特征值图、最小特征值图、R图与最终检测结果展示并存储为图像。

运行简要说明：

1：请安装好python+opencv环境。确保有numpy、scipy等python第三方库。

2：在命令行cmd(windows)/terminal（mac或ubuntu），用cd命令进入hw2目录，运行python harrisCornerDetector.py -file\_path=待处理图片文件路径。

e.g. python harrisCornerDetector.py -file\_path='./3.png'

1. 作业的开发与运行环境

系统版本：macOS Catalina 10.15.1

python版本：3.7.5

opencv版本：4.1.2

python依赖环境：详见hw2文件夹下的requirements.txt

1. 算法的基本原理、流程

基本原理：

Harris Corner Detection算法认为角点应该在窗口的各个方向都有变化，而边界会在某个方向基本不变，而平坦区域在各个方向变化都小。因此计算以下公式：

公式3.1计算了在一个窗口大小内所有点(x, y)在方向[u, v]上的灰度变化。

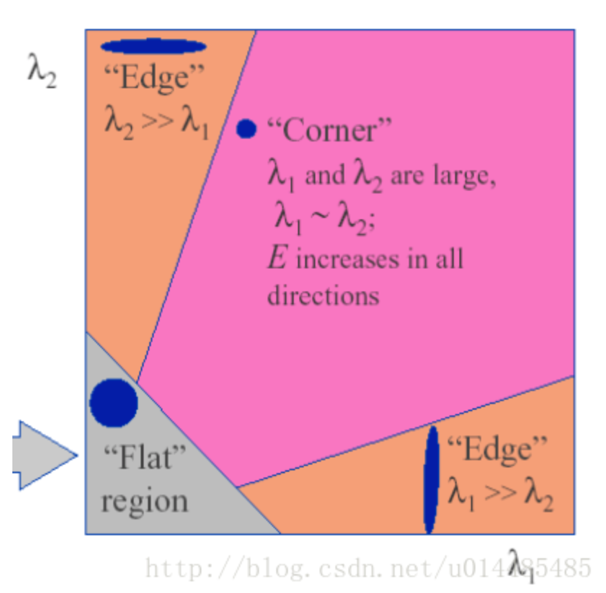
利用二元泰勒公式展开可以得到：

所以：

于是：

记，则

由于角点是在各个方向上变化都大的点，而不是仅沿某个方向[u, v]变化大的点，根据这一特性，考虑M的特征值，点的位置与以下关系：



基于以上原理，Harris Corner Detection定义了一种衡量方法：

R小，说明两个特征值都小，对应区域平坦；

R<0，说明，对应边界；

R大，说明且都大，对应角点；

用点(x,y)的R值是否大于某个阈值判断是否为角点。

主要流程：

|  |
| --- |
| **Input: 图片路径*file\_path***  **Optioal Input: 卷积核大小*kernelSize=3*，窗口大小*blockSize=2*，公式3.6中的*k=0.04*，阈值百分比*threshold=0.01***  **Output: 最大特征值矩阵*𝝀max*，最小特征值矩阵*𝝀min*，响应矩阵*R*，结果图像*resultImg*** |
| # 获得灰度图像  img = ***get\_Gray***(file\_path)  # 得到图像img在x方向与y方向上的导数矩阵Ix, Iy  Ix, Iy = ***get\_Ix\_Iy***(img, kernelSize)  # 遍历图像每个点，通过Ix, Iy获得矩阵M，计算M的特征值𝝀1和𝝀2，最后计算R  **for** Point(x,y) **in** img**:**  M = ***get\_M***(Ix, Iy, x, y, blockSize)  # 计算M的特征值𝝀1和𝝀2  𝝀1, 𝝀2 = ***get\_eig***(M)  𝝀max(x, y) = ***max***(𝝀1, 𝝀2)  𝝀min(x, y) = ***min***(𝝀1, 𝝀2)  R(x, y) = 𝝀1 \* 𝝀2 – k \* (𝝀1 + 𝝀2)^2  # 获得响应矩阵R的最大值  Rmax = R.***max***()  # 获得阈值  cornerThreshold = threshold \* Rmax  # 遍历响应矩阵R，将大于阈值的图像位置标记  resultImg = img.copy()  **for** Point(x, y) **in** R**:**  **if** R(x, y) **is larger than** cornerThreshold**:**  ***Tag*(**resultImg(x, y)) |

1. 具体实现

harrisCornerDetector类：

设计了类harrisCornerDetector来实现Harris角点检测算法。该类具有kernelSize 、k、blockSize、thredshold四个成员变量，并分别具有默认值。

|  |
| --- |
| **def** \_\_init\_\_(self, kernelSize=3, k=0.04, blockSize=2, thredshold=0.01):  self.kernelSize = kernelSize  self.k = k  self.blockSize = blockSize  self.threshold = thredshold |

共设计4个成员方法来实现具体Harris角点检测算法。

方法comput\_Ix\_Iy()：

以灰度图像矩阵为输入，使用opencv自带的Sobel算子计算图像每个点在x方向和y方向上的导数，并返回Ix与Iy。

|  |
| --- |
| **def** computeIx\_Iy(self, img):  Ix = np.zeros(img.shape)  Iy = np.zeros(img.shape)  Ix = cv2.Sobel(img, cv2.CV\_64F, 1, 0, self.kernelSize)  Iy = cv2.Sobel(img, cv2.CV\_64F, 0, 1, self.kernelSize)  **return** (Ix, Iy) |

方法compute\_harris\_response()：

以灰度图像矩阵与文件路径为输入，计算得到最大特征值矩阵𝝀max，最小特征值矩阵𝝀min，响应矩阵R，同时将最大特征矩阵、最小特征矩阵与R以图像形式存储。

|  |
| --- |
| **def** compute\_harris\_response(self, img, path):  Ix, Iy = self.computeIx\_Iy(img)   Mxx = filters.gaussian\_filter(Ix\*Ix, self.blockSize)  Mxy = filters.gaussian\_filter(Ix\*Iy, self.blockSize)  Myy = filters.gaussian\_filter(Iy\*Iy, self.blockSize)   R = np.zeros(img.shape)  height, weight = img.shape  lambda\_max = np.zeros(img.shape)  lambda\_min = np.zeros(img.shape)  **for** row **in** range(height):  **for** col **in** range(weight):  M = np.array([[Mxx[row][col], Mxy[row][col]], [Mxy[row][col], Myy[row][col]]])  l, \_ = np.linalg.eig(M)  lambda\_max[row][col] = l[0] **if** l[0]>l[1] **else** l[1]  lambda\_min[row][col] = l[1] **if** l[0]>l[1] **else** l[0]   cv2.namedWindow(**'lambda\_max'**, cv2.WINDOW\_AUTOSIZE)  egeinMax = lambda\_max.max()  egeinMin = lambda\_min.min()  lambda\_max\_img = (lambda\_max-egeinMin) \* 255. / (egeinMax - egeinMin)  lambda\_min\_img = (lambda\_min - egeinMin) \* 255. / (egeinMax - egeinMin)   lambda\_max\_img = lambda\_max\_img.astype(np.uint8)  lambda\_min\_img = lambda\_min\_img.astype(np.uint8)  cv2.imshow(**'lambda\_max'**, lambda\_max\_img)  cv2.namedWindow(**'lambda\_min'**, cv2.WINDOW\_AUTOSIZE)  cv2.imshow(**'lambda\_min'**, lambda\_min\_img)   *# R = det(M) - k\*trace^2(M)  # det(M) = lambda1 \* lambda2, trace(M) = lambda1 + lambda2  # 韦达定理* detM = Mxx\*Myy-Mxy\*\*2  traceM = Mxx + Myy  R = detM - self.k \* np.power(traceM, 2)   R\_max = R.max()  R\_min = R.min()  cv2.namedWindow(**'R'**, cv2.WINDOW\_AUTOSIZE)  Rimg = (R-R\_min) \* 255. / (R\_max - R\_min)  Rimg = Rimg.astype(np.uint8)  R\_heatmap = cv2.applyColorMap(Rimg, cv2.COLORMAP\_JET)  cv2.imshow(**'R'**, R\_heatmap)   *#write to file* cv2.imwrite(path+**'\_lambda\_max.png'**, lambda\_max\_img)  cv2.imwrite(path+**'\_lambda\_min.png'**, lambda\_min\_img)  cv2.imwrite(path+**'\_R.png'**, R\_heatmap)  **return** R |

使用scipy带的高斯滤波函数filters.gaussian\_filter()计算以blockSize为窗口大小的矩阵Mxx、Mxy、Myy。遍历所有点位置，使用索引建立每个点的矩阵M，使用numpy.linalg.eig()函数计算M的特征值，得到最大特征值矩阵与最小特征值矩阵。

这个方法还存储了角点检测过程中的中间处理结果图像lambda\_max、lambda\_min和R。

由于矩阵M是个2\*2的矩阵,根据线性代数中求解特征值的方法，可以得到：

所以韦达定理可将公式3.6转换为：

为了对最大特征矩阵lambda\_max、最小特征矩阵lambda\_min与R图矩阵R进行可视化，需要对其进行归一化操作，映射到[0, 255]。在这里，使用min-max归一化，即按照公式(4.3)。

使用灰度图展示最大特征图与最小特征图，调用cv2.applyColorMap()使用热力图展示R图。

方法find\_harris\_point():

以灰度图像矩阵和文件路径为输入，找到所有harris角点检测算法检测出来点位置，返回这些位置组成的位置列表。

|  |
| --- |
| **def** find\_harris\_point(self, img, path):  R = self.compute\_harris\_response(img, path)  cornerThreshold = self.threshold \* R.max()  height, weight = img.shape  points = []  **for** row **in** range(height):  **for** col **in** range(weight):  *# the R value of the Point(row, col) is larger than the threshold*  **if** R[row][col] > cornerThreshold:  points.append((row, col))  **return** points |

使用阈值百分比threshold乘以R的最大值作为角点筛选阈值cornerThreshold。遍历所有位置，如果这个位置的R值大于cornerThreshold，则将其添加至返回位置列表points中。

方法plot\_harris\_point():

该方法以图像文件路径为输入，完成harris角点检测并标记角点。

|  |
| --- |
| **def** plot\_harris\_point(self, path):  src = cv2.imread(path)  img = src.copy()  gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)  path\_post = path.split('.')[-1]  path = path[:len(path)-len(path\_post)]  harris\_points = self.find\_harris\_point(gray, path)  **for** point **in** harris\_points:  img[point[0], point[1], :]=[0,0,255]  cv2.namedWindow('resultImg', cv2.WINDOW\_AUTOSIZE)  cv2.imshow('resultImg', img)  cv2.imwrite(path + '\_result.png', img)  key = cv2.waitKey(0)  **if** key == 27:  cv2.destroyAllWindows() |

命令行自定义参数：

为了方便用户使用实现的harris角点检测，使用argparse模块添加命令行自定义参数。

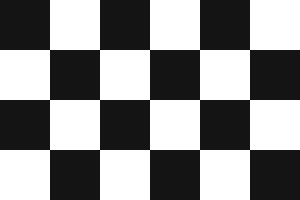
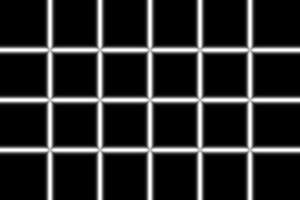
|  |
| --- |
| # add parser  parser = argparse.ArgumentParser(description='HarrisCornerDetector')  parser.add\_argument('-file\_path', type=str, help='待检测的文件路径') |

1. 实验结果与分析
2. 使用课堂中的黑白格子图像

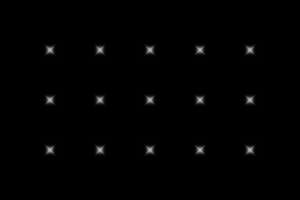
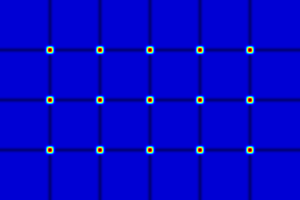
待检测图像见/hw2/3.png。

运行python harrisCornerDetector.py -file\_path='./3.png'。

运行结果见/hw2/3\_result.png、/hw2/3\_lambda\_max.png、/hw2/3\_lambda\_min.png、/hw2/3\_R.png。

/hw2/3\_result.png /hw2/3\_lambda\_max.png

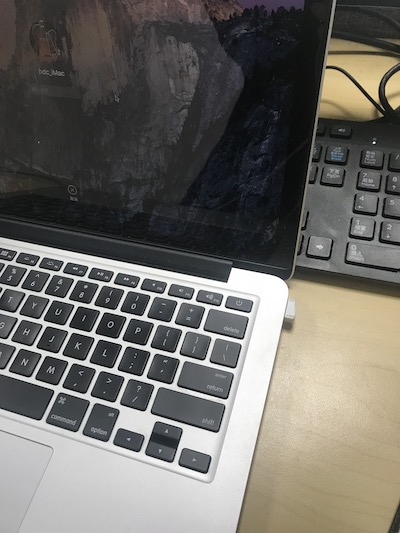
 

/hw2/3\_lambda\_min.png /hw2/3\_R.png

可以看到在不同颜色交点在各个方向上变化，被认为是角点。从/hw2/3\_lambda\_max.png中可以看到不同颜色相交的边缘上最大特征值比较大，而从/hw2/3\_lambda\_min.png这边可以看到边缘上的最小特征值很小，所以只有边缘交集点最大特征值与最小特征值都很大。

1. 拍摄两张对于同一物体在不同光线角度的图片

待检测图像分别为/hw2/pic1.JPG与/hw2/pic2.JPG。

/hw2/pic1.JPG /hw2/pic2.JPG

检测结果分别为/hw2/pic1\_result.png与/hw2/pic2\_result.png。

/hw2/pic1\_result.png /hw2/pic2\_result.png

从以上两张图片中可以看出，尽管在角度和光线发生变化后，对于同一物体其角点大致一致。

1. 结论与心得体会

通过这次作业，对python+opencv如何处理图像特征有了一个比较简单的理解。在计算x方向与y方向上导数时，理解了卷积算子的作用。温故了线性代数的简单求解，简化了R的计算。这次作业由于代码参数名写错了一个字符，导致debug了很久，对自己有点无奈。今后在这方面上要抱以谨慎。此外，感觉到了python对于数据类型检查弱类型的优缺点，在图像数据处理上，数据类型有着比较高的要求，而用python的时候很难对这些情况进行分析。

1. 参考文献

https://blog.csdn.net/u014485485/article/details/79056666