浙江大学

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

一、 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'

二、 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

三、 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：

$$\mathrm{E(u,v)} = \sum_{(x,y)} w(x,y)[I(x+u,y+v) - I(x,y)]^2\,, \qquad \text{公式(3.1)}$$

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：

$$I(x+u,y+v) = I(x,y) + uI_x(x,y) + vI_y(x,y) + O(x,y)\text{，公式(3.2)}$$

所以：

$$I(x+u,y+v) - I(x,y) \approx uI_x + vI_y = [u \quad v]\begin{bmatrix} I_x \\ I_y \end{bmatrix}, \qquad \text{公式(3.3)}$$

于是：

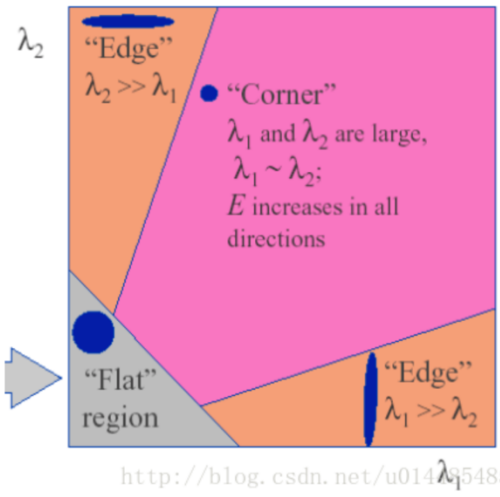
$$E(u,v) = \sum_{(x,y)} w(x,y)[u \quad v]\begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix}\begin{bmatrix} u \\ v \end{bmatrix}, \qquad \text{公式(3.4)}$$

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

$$E(u,v) = [u \quad v]M\begin{bmatrix} u \\ v \end{bmatrix}, \qquad \text{公式(3.5)}$$

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λ_1和λ_2，点的位置与λ_1和λ_2以下关系：



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

$$R = \lambda_1\lambda_2 - k(\lambda_1 + \lambda_2)^2, \qquad \text{公式(3.6)}$$

R 小，说明两个特征值都小，对应区域平坦；

R<0，说明$\lambda_1 \gg \lambda_2$，对应边界；

R 大，说明$\lambda_1 \sim \lambda_2$且都大，对应角点；

用点(x,y)的 R 值是否大于某个阈值θ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))
```

四、 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：

$$\lambda^2 - \left(I_x^2 + I_y^2\right)\lambda + I_x^2 I_y^2 - \left(I_x I_y\right)^2 = 0, \qquad \text{公式(4.1)}$$

所以韦达定理可将公式 3.6 转换为：

$$R = \left(I_x^2 I_y^2 - \left(I_x I_y\right)^2\right) - k\left(I_x^2 + I_y^2\right)^2, \qquad \text{公式(4.2)}$$

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)。

$$Value_{new} = \frac{Value - Value_{min}}{Value_{max} - Value_{min}} \times 255 \qquad \text{公式(4.3)}$$

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')
```

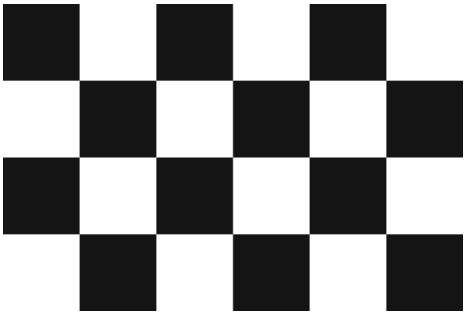
五、 实验结果与分析

1、 使用课堂中的黑白格子图像

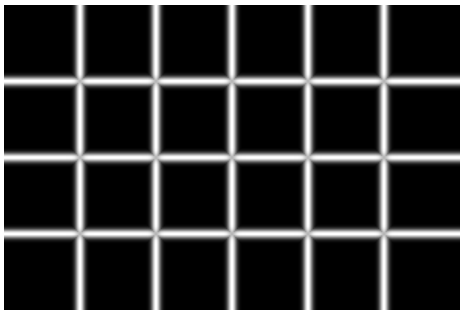
待检测图像见/hw2/3.png。

运行 python harrisCornerDetector.py -file_path='./3.png'。

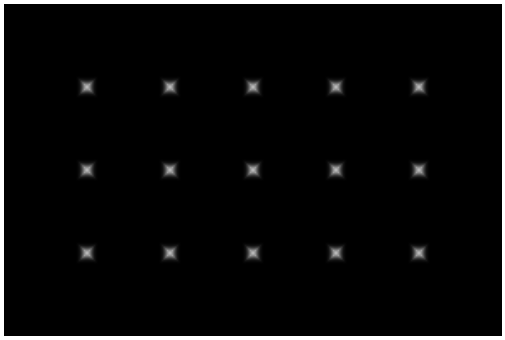
运行结果见/hw2/3_result.png、/hw2/3_lambda_max.png、/hw2/3_lambda_min.png、/hw2/3_R.png。



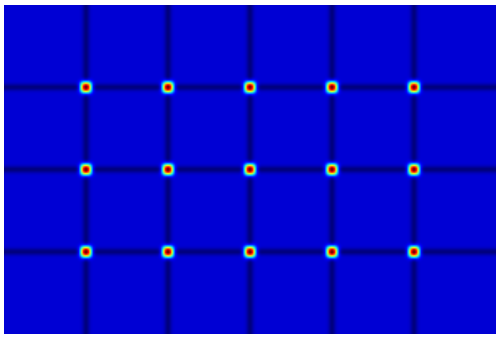
/hw2/3_result.png



/hw2/3_lambda_max.png



/hw2/3_lambda_min.png

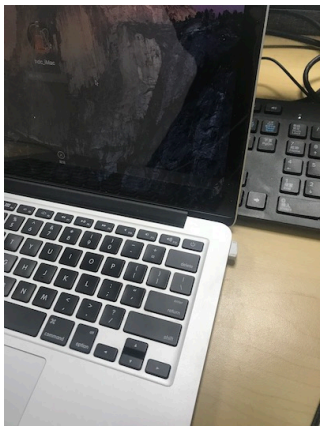


/hw2/3_R.png

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。

2、 拍摄两张对于同一物体在不同光线角度的图片

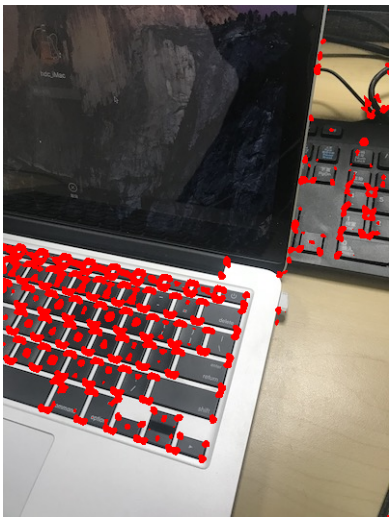
待检测图像分别为/hw2/pic1.JPG 与/hw2/pic2.JPG。



/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。

六、 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。

七、 参考文献

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