

HW03 Harris Corner Detector

3180101041 杨锐

软件开发说明

- @MacOS
- Python == 3.7.6
- Numpy == 1.8.1
- Opencv-python == 4.2.0.34

算法设计说明

Harris Corner Detector常用于提取图像角点和特征推断。与Moravec's corner detector相比，Harris直接考虑了方向，将corner score的差异考虑在内，而不是对每个45度角进行shifting patches，其在区分edges和corners时更准确。

角点检测的思想很简单：当前窗口在所有方向上进行平移，其内的强度变化应该很大。而平滑区域几乎无变化，边缘区域在edge方向上几乎无变化。

因此Harris Corner Detector这样定义一个窗口 w 在 (u, v) 方向平移前后的强度变化：

$$E(u, v) = \sum_{x,y} \underbrace{w(x, y)}_{\text{window function}} \frac{[I(x + u, y + v) - I(x, y)]^2}{\underbrace{I(x, y)}_{\text{intensity}}}$$

为了更好的结果，我们计算高斯卷积，为像素赋予强度值：

```
# Gaussian convolution weights

def Gaussian(p):
    Gaussian_core = []
    for x in range(-1, 2):
        for y in range(-1, 2):
            Gaussian_core.append(1. / (2 * np.pi * (p ** 2)) *
                                 np.e **(-(x ** 2 + y ** 2) / (2 * (p ** 2))))
    return Gaussian_core
```

对于corners， $E(u, v)$ 的结果应该尽可能大，所以先经过泰勒展开得到：

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

其中：

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

其中: (I_x, I_y) 分别对应图像 (x, y) 方向上的梯度。

接下来我们需要根据 M 区分出flat, edges, corners:

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

其中:

$$\begin{aligned}\det(M) &= \lambda_1 \lambda_2 \\ \text{trace}(M) &= \lambda_1 + \lambda_2 \\ (\lambda_1, \lambda_2) &\text{ 对应 } M \text{ 的特征值}\end{aligned}$$

因此我们便得到了判断机制:

- $|R|$ 小, 说明 (λ_1, λ_2) 都很小, flat region
- $R < 0$, 说明 $\lambda_1 \gg \lambda_2$ 或 $\lambda_1 \ll \lambda_2$, edges region
- R 很大, 说明 λ_1, λ_2 都很大, corner region

至此我们便得到了整个检测过程的算法, 下面是具体代码实现:

第一步: 转换图片并进行原始数据计算, 初始化数据结构

```
# to gray
height, width, _ = img.shape
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Sobel operator calculates the gradient
x_sobel = cv2.Sobel(gray, cv2.CV_64F, 1, 0, ksize=5)
y_sobel = cv2.Sobel(gray, cv2.CV_64F, 0, 1, ksize=5)

# initialize to store medium result
max_show = np.zeros((height-2, width-2, 3), np.uint8)
min_show = np.zeros((height-2, width-2, 3), np.uint8)
R_show = np.zeros((height-2, width-2, 3), np.uint8)

max_list = []
min_list = []
R_list = []
k = 0.05 # Constant coefficient
Gaussian_core = Gaussian((size-1)/6)
```

第二步: 遍历窗口, 计算 M

```

for x in range(1, width-1):
    for y in range(1, height-1):
        M = np.mat([[0., 0.], [0., 0.]])
        for i in range(x-1, x+2):
            for j in range(y-1, y+2):
                M += Gaussian_core([(i-(x-1))*(j-(y-1))+(j-(y-1))] * np.mat(
                    [[(x_sobel[j][i])**2, x_sobel[j][i]*y_sobel[j][i]], [x_sobel[j]
[i]*y_sobel[j][i], (y_sobel[y][x])**2]])
        lamda, _ = linalg.eig(M)
        lamda1 = max(lamda)
        lamda2 = min(lamda)
        max_list.append(lamda1)
        min_list.append(lamda2)
        R_list.append(lamda1*lamda2-k*(lamda1+lamda2)**2)

```

第三步：非最大值抑制，筛选满足条件的R

```

for x in range(0, width-2):
    for y in range(0, height-2):
        record1 = int((max_list[x * (height - 2) + y] / normalize_1) * 255)
        record2 = int((min_list[x * (height - 2) + y] / normalize_2) * 255)
        record_R = int((R_list[x * (height - 2) + y] / normalize_R) * 255)
        max_show[y][x] = (record1, record1, record1)
        min_show[y][x] = (record2, record2, record2)
        if(R_list[x * (height - 2) + y] > 0.005*normalize_R and
non_maximum_suppression(R_list, x, y, 3, width-2, height-2)):
            cv2.circle(img, (x+1, y+1), 1, (0, 0, 255), -1)
        R_show[y][x] = (record_R, record_R, record_R)

```

至此，我们便得到所有满足条件的角点。

最后，将中间结果和最终结果保存。

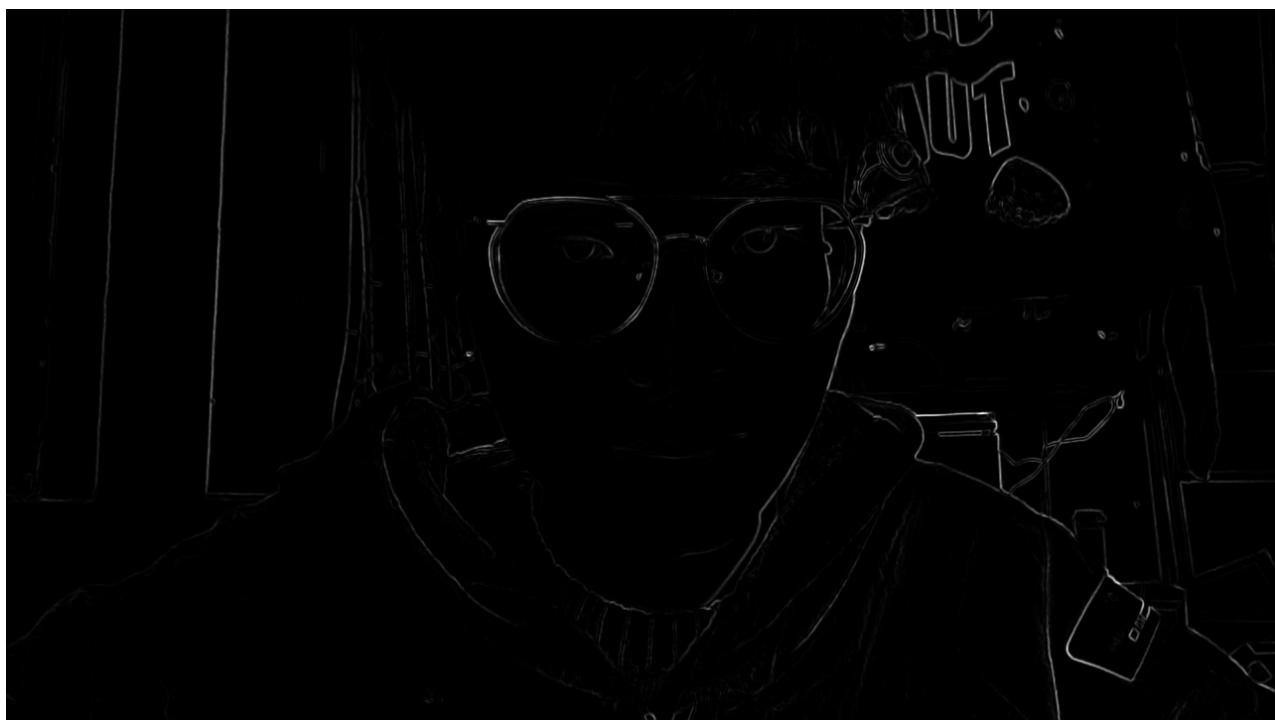
实验结果分析

原始图像



frame

最大特征值



Max

最小特征值



Min

R



R

最终结果



Haris

编程体会

- 掌握了Harris Corner Detection

个人照片



