

第 15 课 图像处理——角点检测

1.角点介绍

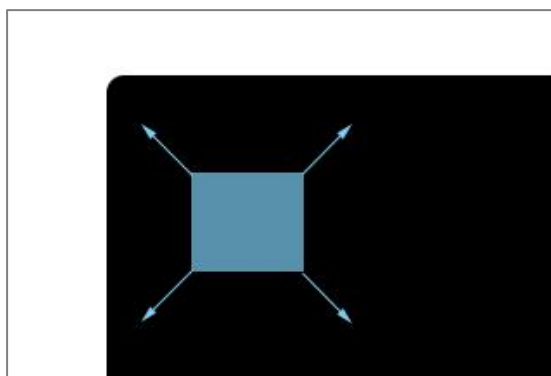
1.1 角点定义

角点的定义有两个版本：一是两条边缘的交点，二是邻域内具有两个主方向的特征点。一般而言，角点是边缘曲线上曲率为极大值的点，或者图像亮度发生剧烈变化的点。

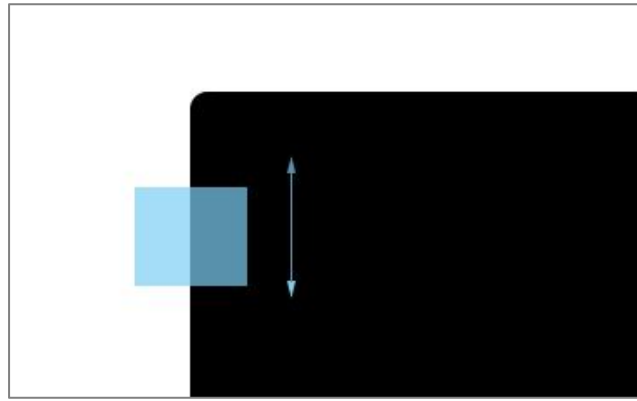
1.2 检测思路

在图像中定义一个局部小窗口，然后沿各个方向移动这个窗口，则会出现三种情况，分别对应平坦区、边缘和角点。

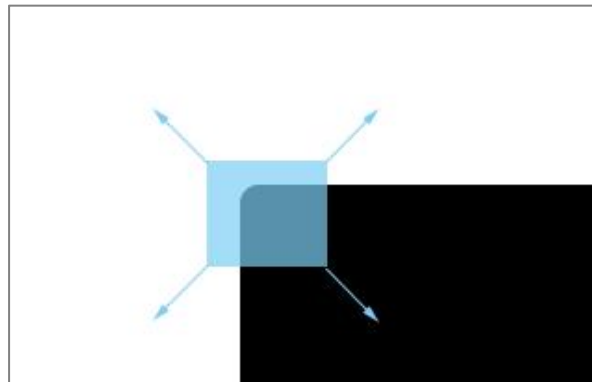
- 1) 窗口内的图像强度，在窗口向各个方向移动时，都没有发生变化，则窗口内都是“平坦区”，不存在角点。



- 2) 窗口内的图像强度，在窗口向某一个 (些) 方向移动时，发生较大变化；而在另一些方向不发生变化，那么窗口内可能存在“边缘”。



3) 窗口内的图像强度，在窗口向各个方向移动时，都发生了较大的变化，则认为窗口内存在“角点”。



1.3 Harris 角点检测公式推导

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

其中：

- 1) $\det(M) = \lambda_1 \lambda_2$ 。
- 2) $\text{trace}(M) = \lambda_1 + \lambda_2$ 。
- 3) λ_1 和 λ_2 是矩阵 M 的特征值。

我们根据这些特征来判断一个区域：当 λ_1 和 λ_2 都小时， $|R|$ 也小，则为平面区域。当 $\lambda_1 \gg \lambda_2$ 或者 $\lambda_1 \ll \lambda_2$ 时， R 小于 0，则为边缘。当 λ_1 和 λ_2 都很大时，且 $\lambda_1 \sim \lambda_2$ 中时， R 也很大，(λ_1 和 λ_2 中的最小值都大于阈值)，则为角点。

2.实验步骤



接下来我们通过 Harris 角点检测，检测出图像的角点区域。

注意：

1) 需要先将目录“第四章 OpenCV 计算机视觉学习->图像处理进阶篇->第 15 课 图像处理——角点检测->例程源码”下的例程“corners_demo.py”和示例图片“test.jpg”复制到共享文件夹。

2) 共享文件夹的配置方法可查看目录“第 2 章 Linux 系统简介及使用入门->Linux 基础课程->第 3 课 Linux 系统安装及换源方法”下的文档。

3) 输入指令时需严格区分大小写，且可以使用“Tab”键补齐关键字。

1) 打开虚拟机，启动系统。点击系统任务栏的图标, 并点击图标, 或使用快捷键“Ctrl+Alt+T”，打开命令行终端。

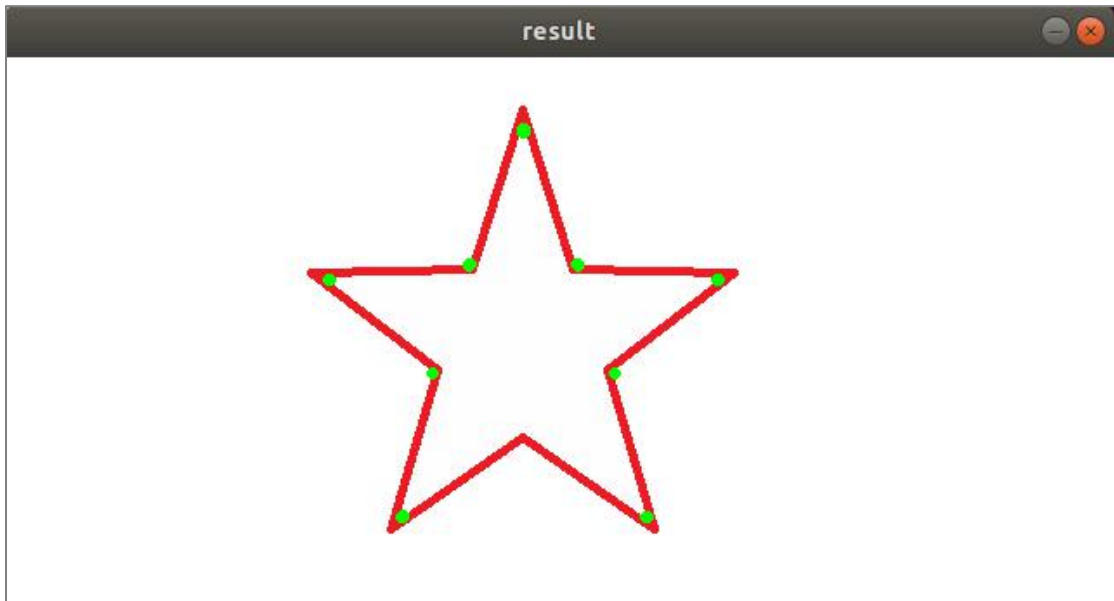
2) 输入指令“`cd /mnt/hgfs/share/`”，并按下回车，进入共享文件夹。

```
hiwonder@ubuntu:~$ cd /mnt/hgfs/Share/
```

3) 输入指令“`python3 corners_demo.py`”，并按下回车，运行例程。

```
ubuntu@ubuntu-virtual-machine:/mnt/hgfs/share$ python3 corners_demo.py
```

3.实现效果



4.代码分析

可在目录“第四章 OpenCV 计算机视觉学习->图像处理进阶篇->第 15 课 图像处理——角点检测->例程源码”下查看例程“corners_demo.py”。

```
import numpy as np
import cv2 as cv

def harris(image):
    # Detector parameters
    blockSize = 2
    apertureSize = 3
    k = 0.04
    # Detecting corners
    gray = cv.cvtColor(image, cv.COLOR_BGR2GRAY)
    dst = cv.cornerHarris(gray, blockSize, apertureSize, k)
    # Normalizing
    dst_norm = np.empty(dst.shape, dtype=np.float32)
    cv.normalize(dst, dst_norm, alpha=0, beta=255, norm_type=cv.NORM_MINMAX)
    # Drawing a circle around corners
    for i in range(dst_norm.shape[0]):
        for j in range(dst_norm.shape[1]):
            if int(dst_norm[i, j]) > 120:
                cv.circle(image, (j, i), 2, (0, 255, 0), 2)
    # output
    return image

src = cv.imread("test.jpg")
result = harris(src)
cv.imshow('result', result)
cv.waitKey(0)
cv.destroyAllWindows()
```

1) 导入模块：导入 cv2 模块和 numpy 模块。

```
gray = cv.cvtColor(image, cv.COLOR_BGR2GRAY)
```

2) 创建角点检测处理函数: **harris(image)**: 参数为需处理的图像。

3) 颜色空间转换: **cvtColor(src,mode)**

第一个参数 “**src**”, 是转换的图像。

第二个参数 “**mode**”, 是颜色空间的转换模式。

```
dst = cv.cornerHarris(gray, blockSize, apertureSize, k)
```

4) 角点检测: **cornerHarris(src,blockSize,apertureSize,k)**

第一个参数 “**src**”, 是检测的图像。

第二个参数 “**blockSize**”, 是角点检测中领域像素的大小。

第三个参数 “**apertureSize**”, 求导中使用的窗口大小。

第四个参数 “**k**”, 自由参数, 取值范围为[0.04, 0.06]。

```
dst_norm = np.empty(dst.shape, dtype=np.float32)
```

5) 获取同类型数组: **empty(shape,dtype)**

第一个参数 “**shape**”, 整数或者整型元组定义的返回数组的形状。

第二个参数 “**dtype**”, 数据类型, 定义返回数组的类型。

```
cv.normalize(dst, dst_norm, alpha=0, beta=255, norm_type=cv.NORM_MINMAX)
```

6) 归一化处理: 归一化就是要把需要处理的数据经过处理后限制在你需要的一定范围内, 经过归一化转化的 Harris 角点检测结果图像, 大小与原图像一致, 每一个点的像素值大小对应的是原图像上该点是角点的概率, 值越大, 越有可能是角点。

函数格式: **normalize(src,dst,alpha,beta,normType)**

第一个参数 “**src**”, 是输入的数组。

第二个参数 “**dst**”, 是处理后输出的数组。

第三个参数 “**alpha**”, 是归一化处理的最小值。

第四个参数“**beta**”，是归一化处理的最大值。

第五个参数“**normType**”，是归一化的类型，具体如下：

1) **NORM_MINMAX**: 数组的数值被平移或缩放到一个指定的范围，线性归一化，一般较常用。

2) **NORM_INF**: 是归一化数组的 C-范数(绝对值的最大值)

3) **NORM_L1**: 归一化数组的 L1-范数(绝对值的和)

4) **NORM_L2**: 归一化数组的(欧几里德)L2-范数

```
for i in range(dst_norm.shape[0]):
    for j in range(dst_norm.shape[1]):
        if int(dst_norm[i, j]) > 120:
            cv.circle(image, (j, i), 2, (0, 255, 0), 2)
```

7) **圈出角点**: 用循环遍历归一化后的图像数组，在角点区域画圆圈出。

画圆函数格式: **circle(src,point,radius,color,thickness)**

第一个参数“**src**”，是需要绘制的图像。

第二个参数“**point**”，是绘制的圆心。

第三个参数“**radius**”，是绘制圆的半径。

第四个参数“**color**”，是设置的颜色。

第五个参数“**thickness**”，是圆的线条宽度，负数为实心圆。

```
src = cv.imread("test.jpg")
result = harris(src)
cv.imshow('result', result)
cv.waitKey(0)
cv.destroyAllWindows()
```

8) **读取图像，处理图像，显示图像和关闭窗口**:

读取图像: 使用 **imread(image)**函数读取图像，参数为图像名称。

处理图像: 调用 **harris(image)**函数处理图像，传入处理的图像参数。

显示图像: 使用 **imshow(title,src)**函数显示处理结果图像，参数是窗口的标题和显示的

图像。

关闭窗口：使用 `waitKey` 函数等待按键按下，再调用 **`destroyAllWindows`** 关闭显示窗口。