浙江大学

计算机视觉(本科)作业报告

作业名称： Harris Corner Detection

姓　　名：

学　　号：

电子邮箱：

联系电话：

导　　师：

2019 年 12 月 9 日

Harris Corner Detection

一、 作业已实现的功能简述及运行简要说明

1.功能简述

本次实验的要求是对输入的一张彩色图片实现 Harris Corner 检测算法。通过①将图片转换为灰度图像；②用高斯滤波器平滑图像；③计算图像 X 方向与 Y 方向的一阶高斯偏导数及其相关矩阵；④得到 M 矩阵，计算矩阵的两个特征值，得到矩阵的最大特征值矩阵和最小特征值矩阵，计算得到 R；⑤根据最大特征值矩阵和最小特征值矩阵得到图像的最大特征值图和最小特征值图，给 R 设置阈值得到图像的边和角点图；⑥根据 R 得到 R 图；⑦筛选检测结果，在原图上叠加检测结果。

其中，在本实验过程中，没有调用 OPENCV 或其他 SDK 里与 Harris 角点检测相关的函数；最大特征值图，最小特征值图，R 图，原图上叠加检测结果图，图像的边、角点图等在实验结果中展示。

在本次实验中，最后可以得到中间的处理结果及最终的检测结果，输出成图像文件，包括最大特征值图，最小特征值图，R 图，原图上叠加检测结果图，图像的边、角点图。

2.运行简要说明：

打开 Pycharm 运行程序，可以在运行程序过程中显示视频，并在./code/img 文件夹下生成 6 个 jpg 文件，分别为最大特征值图，最小特征值图，R 图，原图上叠加检测结果的最终结果图，图像的边图和角点图。

二、 作业的开发与运行环境

Windows 10

Python 3.7.4

Opencv 4.1.1

三、 系统或算法的基本思路、原理、及流程或步骤等

程序分为以下七个步骤：

1. 将图片转换为灰度图像

读入图像，得到原始图像矩阵，将图像转换为灰度图像，并记录图像大小。

2.用高斯滤波器平滑图像

对上一步得到的灰度图像进行高斯平滑。

3. 计算图像 X 方向与 Y 方向的一阶高斯偏导数及其相关矩阵

运用 Sobel 算子计算 X 方向与 Y 方向的一阶高斯偏导数，进行运算得到 Ix^2，Iy^2

和 IxIy，再对 Ix^2，Iy^2和 IxIy 进行高斯卷积得到 Sxx，Syy 和 Sxy。

4. 得到 M 矩阵计算矩阵的两个特征值，得到矩阵的最大特征值矩阵和最小特征值矩阵，计算得到 R

根据上一步得到的矩阵算出 M 矩阵，由此得到矩阵的两个特征值，根据两个特征值计算 R，并由此得到最大特征值矩阵和最小特征值矩阵。

5. 根据最大特征值矩阵和最小特征值矩阵得到图像的最大特征值图和最小特征值图，给 R 设置阈值得到图像的边和角点图

对最大特征值矩阵和最小特征值矩阵进行归一化操作，将处理后的矩阵写成图像并保存成最大特征值图和最小特征值图，给 R 设置阈值，取一定范围内的点，分别得到图像的边和角点图。

6. 根据 R 得到 R 图

对 R 归一化处理后，根据处理后的矩阵得到 R 图。

7. 筛选检测结果，在原图上叠加检测结果

根据矩阵制定合适的 mask，保留 mask 内最大的值点，其它设置为 0，并将该点绘制在原图上，得到叠加后的检测结果图。

四、 具体如何实现，例如关键（伪）代码、主要用到函数与算法等（为了表现清晰，在这里放置一组结果图，更多结果图会在实验结果中展示）

1. 将图片转换为灰度图像

（1）首先读入图像，得到原始图像矩阵，这里运用了 cv2. imread(filename, flags)函数，其中 filename 为导入图片的路径文件名。

（2）将图像转换为灰度图像，这里运用了 cv2.cvtColor(src, code, dst, dstCn)函数，其中 src 为需要转换的图像，code 为颜色映射类型，在程序中我运用到的实例为：cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)。

（3）记录图像大小，即用 h,w 来代表图像大小。

2. 用高斯滤波器平滑图像

这里用到了 cv2.GaussianBlur(src, ksize, sigmaX, dst=None, sigmaY=None, borderType=None)函数，其中 src 是需要进行高斯平滑的图像，ksize 是高斯核的大小，即（width, height），两者都是正奇数（如果设为 0 的话可以根据 sigma 得到），sigmaX 为 X 方向的高斯核标准差，sigmaY 为 Y 方向的高斯核标准差。在这里我的应用实例为 cv2.GaussianBlur(gray, (3, 3), 0)。

3. 计算图像 X 方向与 Y 方向的一阶高斯偏导数及其相关矩阵

（1）运用 Sobel 算子计算 X 方向与 Y 方向的一阶高斯偏导数，这里运用的函数为 Sobel(src, ddepth, dx, dy, dst=None, ksize=None, scale=None, delta=None,

borderType=None)函数，其中 src 指需要进行运算的矩阵，ddepth 指图像的深度，-1 表示的是与原图像相同的深度，dx 和 dy 表示的是求导的阶数，0 表示这个方向上没有求导，ksize 表示 Sobel 算子的大小，必须为 1、3、5、7。这里我的应用实例为：用 cv2.Sobel(gray, -1, 1, 0, ksize=3)进行 x 方向的求导，用 cv2.Sobel(gray, -1, 0, 1, ksize=3)进行 y 方向的求导

（2）进行运算得到 Ix2，Iy2 和 IxIy，即对 Ix 和 Iy 进行乘法运算。

（3）再对 Ix2，Iy2 和 IxIy 进行高斯卷积得到 Sxx，Syy 和 Sxy，这里运用到的高斯卷积算子为（Fig-1）：

$$\frac{1}{16}\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix}$$

Fig-1

进行卷积运用的函数为 filter2D(src, ddepth, kernel, dst=None, anchor=None, delta=None, borderType=None)，其中，src 指原矩阵，ddepth 指目标所需深度，kernal 指卷积核，这里用到的实例举例：Sxx = cv2.filter2D(Ix2, -1, kernal)。

4. 得到 M 矩阵计算矩阵的两个特征值，得到矩阵的最大特征值矩阵和最小特征值矩阵，计算得到 R

（1）根据上一步得到的矩阵算出 M 矩阵，以下分别是 M 原矩阵所对应的值（Fig-2）和用我函数中所得到的变量来表示 M（Fig-3）。

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

Fig-2

$$M=\begin{bmatrix}\mathrm{Sxx} & \mathrm{Sxy}\\ \mathrm{Sxy} & \mathrm{Syy}\end{bmatrix}$$

Fig-3

得到 M 矩阵的两个特征值，这里我运用的函数为 numpy.linalg.eig(a)，其中参数 a 为想要计算特征值的方阵，返回值为 w（特征值）和 v（特征向量），对两个特征值进行大小比较后得到最大特征值矩阵和最小特征值矩阵。

计算 M 矩阵和 M 的特征值用代码表示如下：

```
for i in range(h):
    for j in range(w):
        M[0, 0] = Sxx[i, j]
        M[0, 1] = Sxy[i, j]
        M[1, 0] = Sxy[i, j]
        M[1, 1] = Syy[i, j]

        #计算 M 的特征值
        e, f = np.linalg.eig(M)
        [lam1, lam2] = e
```

根据两个特征值计算 R，R 的计算公式（Fig-4）中的 λ1 和 λ2 分别为矩阵 M 的两个特征值。

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

$$\det M = \lambda_1 \lambda_2$$
$$\operatorname{trace} M = \lambda_1 + \lambda_2$$

$$(k - \text{empirical constant},\ k = 0.04\text{-}0.06)$$

Fig-4

计算 R 的过程用代码表示如下：

```
#计算 R
detM = lam1 * lam2
traceM = lam1 + lam2
R[i, j] = detM - 0.05 * (traceM ** 2)
```

5. 根据最大特征值矩阵和最小特征值矩阵得到图像的最大特征值图和最小特征值图，给 R 设置阈值得到图像的边和角点图

对最大特征值矩阵和最小特征值矩阵进行归一化操作，进行归一化操作我用到的函数为 `cv2.normalize(src, dst, alpha=None, beta=None, norm_type=None, dtype=None, mask=None)`，其中 src 代表输入数组，dst 表示与 src 大小相同的输出数组。我用到的归一化方式为 NORM_MINMAX，即数组的数值被平移或缩放到一个指定的范围（这里我设置的范围为 0 到 255）线性归一化，这里我用到的函数实例举例为 `cv2.normalize(lam_max, max1, 0, 255, cv2.NORM_MINMAX)`。将处理后的矩阵写成图像并保存成最大特征值图和最小特征值图。

给 R 设置阈值，取一定范围内的点，分别得到图像的边和角点图。这里我分别

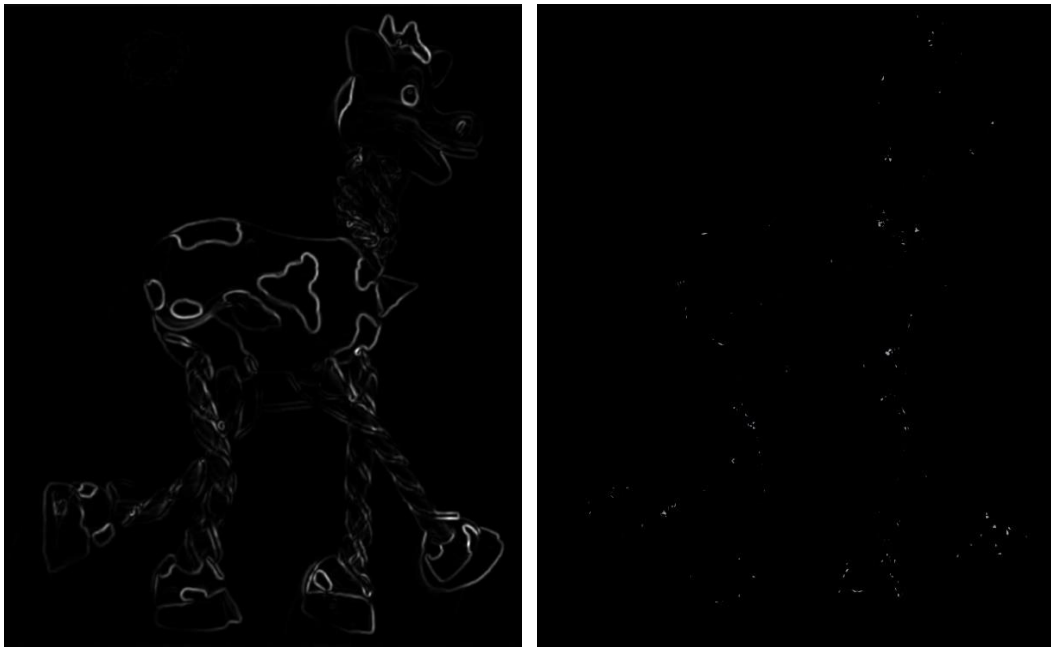
numpy.max()函数和 numpy.min()函数得到矩阵的最大值和最小值，设置阈值，当 R 值大于 0，且大于最大值的 0.001 倍时，将此点设置为角点，最后制作成一张图像的角点图。当 R 值小于零且其绝对值大于矩阵最小值的绝对值的 0.001 倍时，将此点保留，最后制作成一张图像的边图。

得到图像的边图和角点图的代码如下：

```
# 设置阈值
#得到图像的角点
for i in range(h):
    for j in range(w):
        if R[i,j] > 0:
            if R[i,j] > max * 0.001:
                corner[i,j] = 255

#得到图像的边
for i in range(h):
    for j in range(w):
        if R[i,j] < 0:
            if abs(R[i,j]) > abs(min*0.001):
                edge[i,j] = 255
```

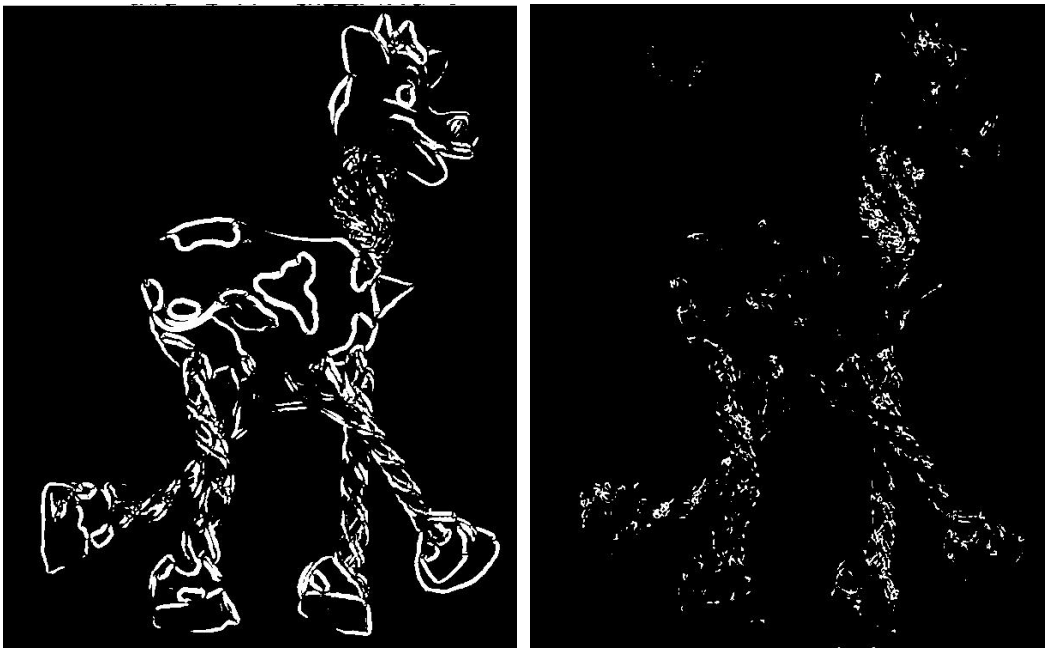
如 Fig-5 所示，最大特征值图和最小特征值图可以显示出一定的效果。



最大特征值图　　最小特征值图

Fig-5

如 Fig-6 所示，边图和角点图可以显示出一定的效果。



边图　　角点图

Fig-6

6. 根据 R 得到 R 图

对 R 归一化处理后，根据处理后的矩阵得到 R 图。在归一化操作中，同样采取 `cv2.normalize(src, dst, alpha=None, beta=None, norm_type=None, dtype=None, mask=None)`函数，用到的归一化方式仍为 NORM_MINMAX，即数组的数值被平移或缩放到一个指定的范围（这里我设置的范围为 0 到 255）线性归一化，这里我用到的函数实例举例为 `cv2.normalize(R,n_R,0,255,cv2.NORM_MINMAX)`。

之后将归一化操作后的图转为色彩图，这里用到的函数为 `cv2.applyColorMap (src, colormap, dst=None)`，其中 src 表示要进行转化的灰度图，colormap 表示选取的颜色图，这里我应用到的实例为 `heat_img = cv2.applyColorMap(n_R, cv2.COLORMAP_JET)`，此处的三通道热力图是 cv2 专有的 BGR 排列，之后将 BGR 图像转为 RGB 图像，运用 `cv2.cvtColor(src, code, dst, dstCn)`函数，其中 src 为需要转换的图像，code 为颜色映射类型，在程序中我运用到的实例为 `heat_img = cv2.cvtColor(heat_img, cv2.COLOR_BGR2RGB)` 。

得到 R 图的过程的过程用代码表示如下：

```
1.  n_R = np.zeros([h,w])
2.  cv2.normalize(R,n_R,0,255,cv2.NORM_MINMAX)
3.  n_R = n_R.astype(np.uint8)
4.  heat_img = cv2.applyColorMap(n_R, cv2.COLORMAP_JET)  # 此处的三通道热力图是 cv2 专有的 BGR 排列
5.  heat_img = cv2.cvtColor(heat_img, cv2.COLOR_BGR2RGB)  # 将 BGR 图像转为 RGB 图像
```

仍然拿第一张小马的照片做测试，得到的 R 图如下（如 Fig-7 所示）。

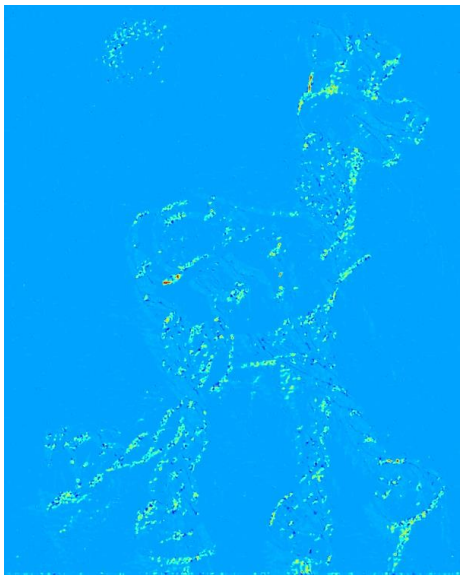


Fig-7

7. 筛选检测结果，在原图上叠加检测结果

根据矩阵制定合适的 mask，使用 15*15 的 mask，用 mask 扫描图像，保留图像在 mask 矩阵中的最大的值点，其它设置为 0，并调用 `cv2.circle(img,center,radius, color,thickness=None, lineType=None, shift=None)`函数画圆，其中 `img` 代表需要绘制圆的图片，`center` 代表圆心位置，`radius` 代表半径，`color` 代表圆的颜色，我们用此函数绘制以最大值点为圆心，1 为半径的圆将该点绘制在原图上，得到叠加后的检测结果图。

叠加检测结果后的图如 Fig-8 所示。

Fig-8

五、 实验结果与分析

我所选用的五张图像如下：



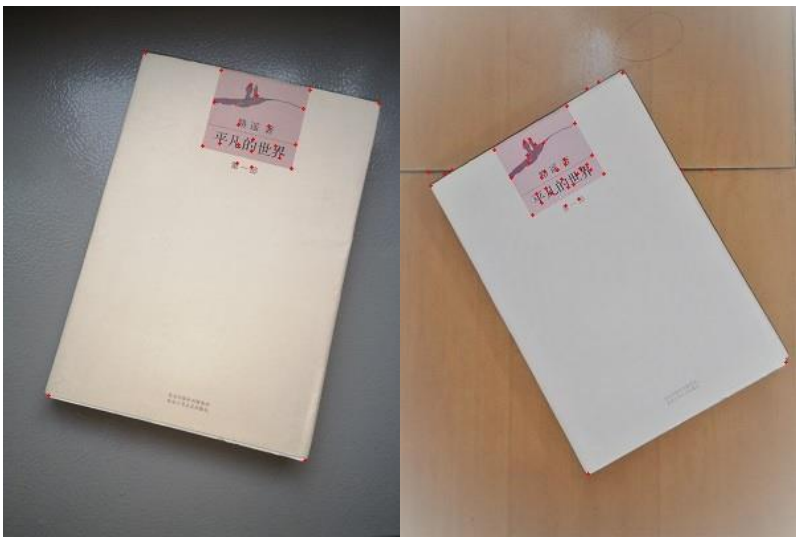


1.相同物体在不同背景或灯光下的对比

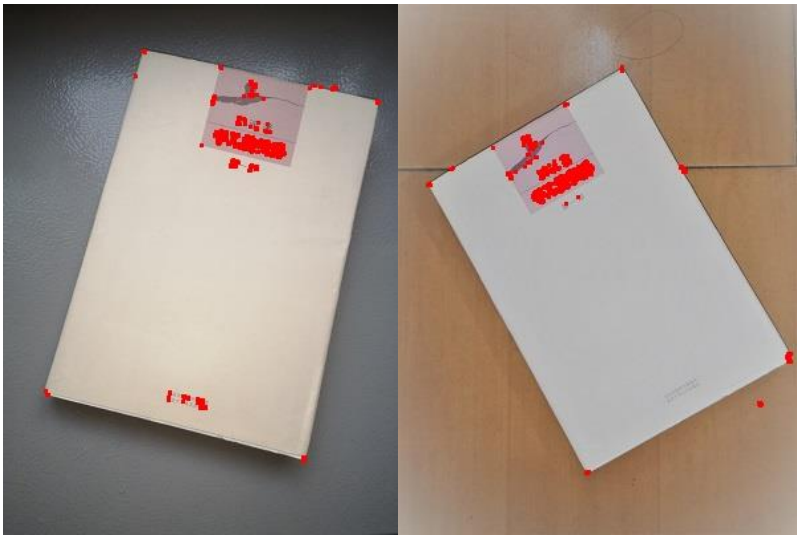
（1）horse 图像

与 OpenCV 自带函数进行对比：

（1）book 图像



与 OpenCV 自带函数进行对比：

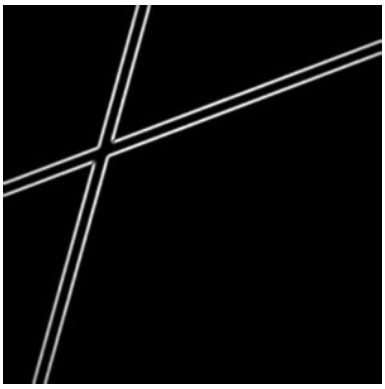


由上面的结果来看，在最后的叠加检测结果上，与 OPENCV 自带的函数对比来看，我们可以发现 OpenCV 对于某一些点的检测更准一些。在自己实现的算法中相同物体在不同位置的两张图片中有很大一部分的检测结果是相对应的，但是同时在马的脚上也有一些是不对应的，总体上看检测算法是有一定的效果的。

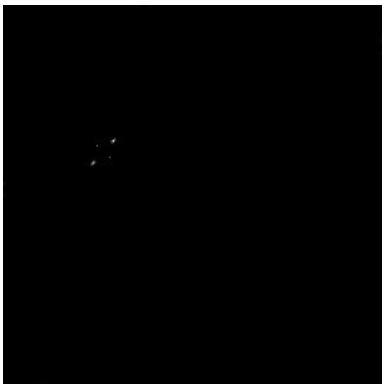
2. 所有图片的过程及结果图

（1）line 图像

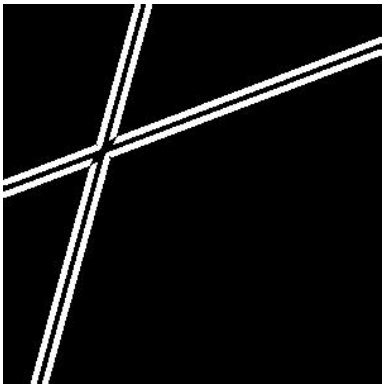
最大特征值图



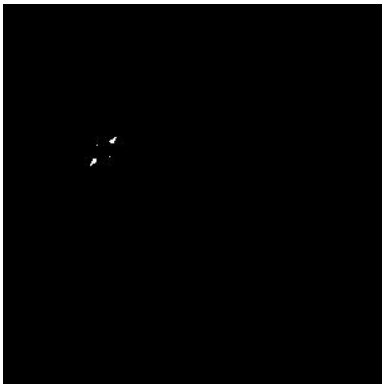
最小特征值图



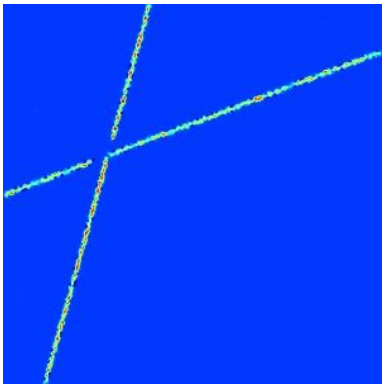
图像的边图



图像的角点图



R 图



原图上叠加检测结果图

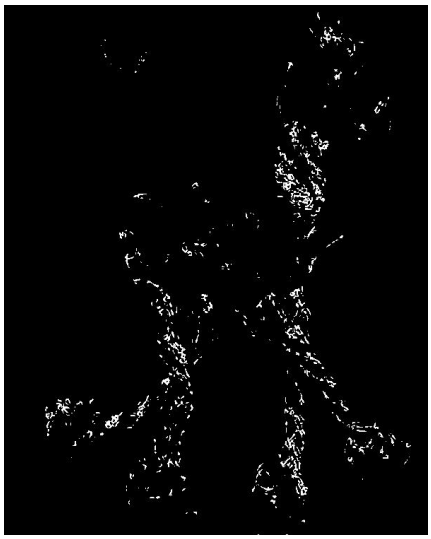
（2）horse 图像 1

最大特征值图

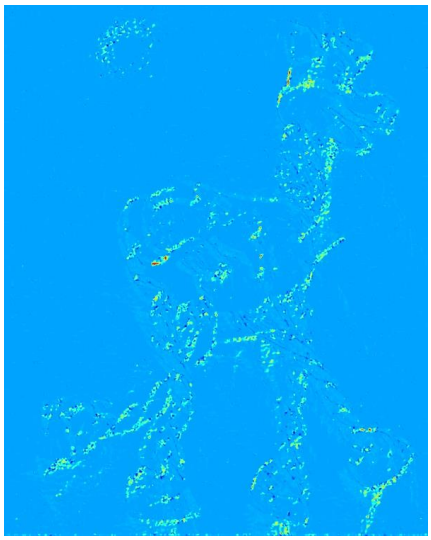
最小特征值图

图像的边图

图像的角点图



R 图

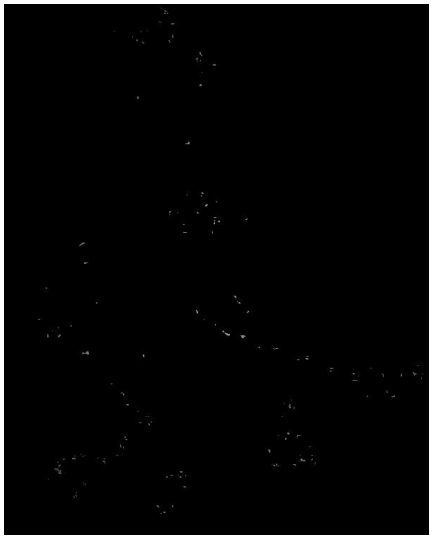


原图上叠加检测结果图

（3）horse 图像 2

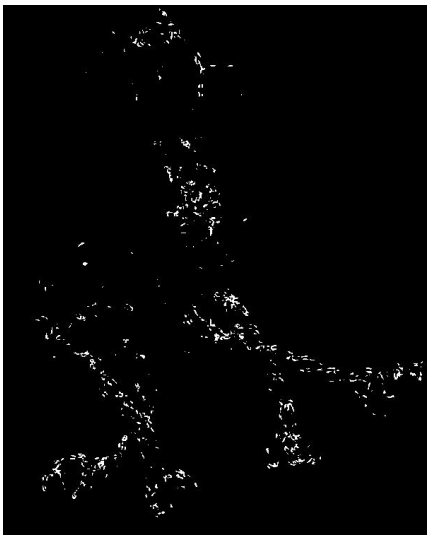
最大特征值图

最小特征值图



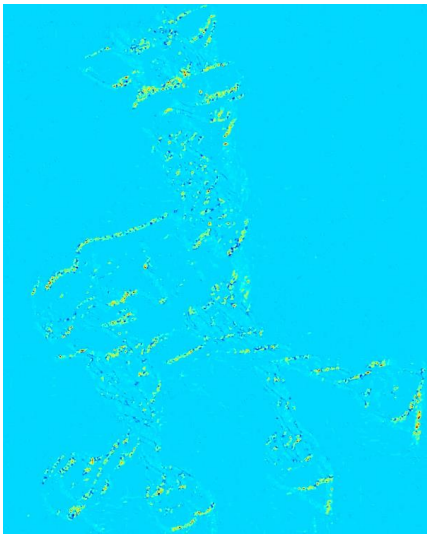
图像的边图

图像的角点图



R 图

原图上叠加检测结果图

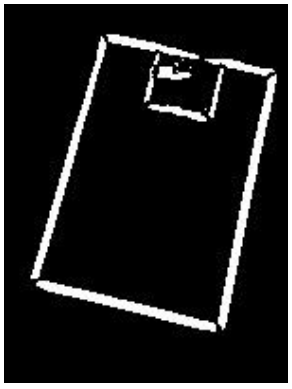


（4）book 图像 1

最大特征值图

最小特征值图

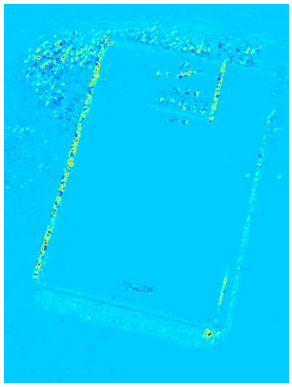
图像的边图



图像的角点图

R 图

原图上叠加检测结果图

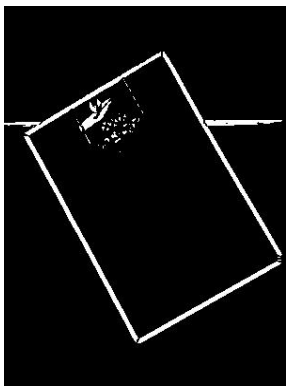


（5）book 图像 2

最大特征值图

最小特征值图

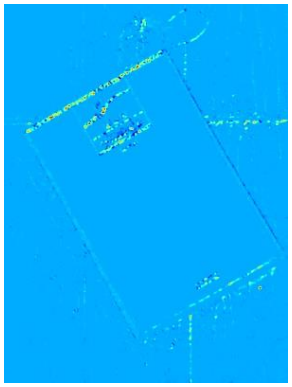
图像的边图



图像的角点图

R 图

原图上叠加检测结果图

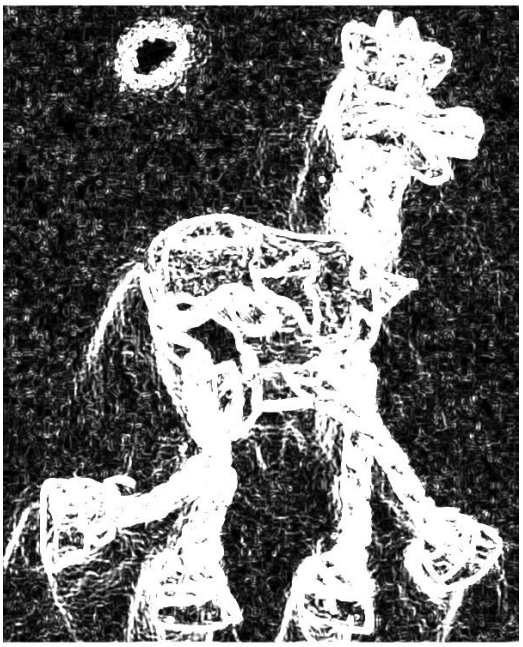


由上面的结果来看，本次实验的 Harris Corner Detection 取得了一些成果。

六、 结论与心得体会

在完成本次实验的过程中主要遇到了两个问题，接下来将主要介绍这两个问题及其解决方案。

第一个问题出现在最大特征值图和最小特征值图的过程中，由于刚开始时并没有进行对于直接赋值之后的最大特征值矩阵和最小特征值矩阵的归一化，刚开始得到的最大特征值图和最小特征值图非常奇怪。以最大特征值图为例，未归一化和归一化后的图对比如下：



第二个问题出现在最后在原图上叠加检测结果的图上，由于刚开始选择的 mask 矩阵过小，导致得到了很多不必要的点，在后面通过不断修改 mask 大小后得到了比较好的效果。改变前后的差别如下：


通过本次实验，我了解了 Harris Corner Detection 算法，在动手做的过程发现并解决问题，在不断的思考和改正中完成了本次作业，同时巩固了 Python 的语法及运用，感觉收获很大，受益匪浅。

七、 参考文献

Sobel 算子：https://blog.csdn.net/huanhuanq1209/article/details/79845061

filter2D 函数：https://www.cnblogs.com/lfri/p/10599420.html

特征值函数：https://blog.csdn.net/rainpasttime/article/details/79837368

归一化函数：https://blog.csdn.net/qq_29023939/article/details/81105806