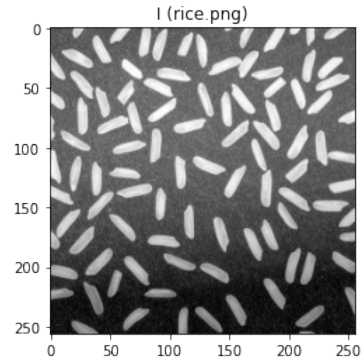
《图像分析与理解》课程——图像配准相关实验

212138 卓旭

Part 1 – Harris 角点检测

1. 读取 rice.png，结果如下：

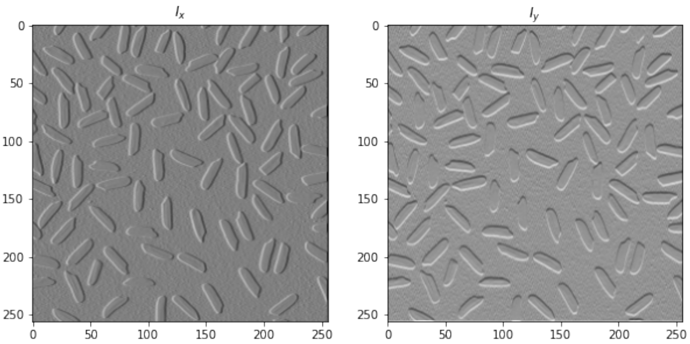


2. 计算每个像素的梯度I_x, I_y形成梯度图。按照梯度定义

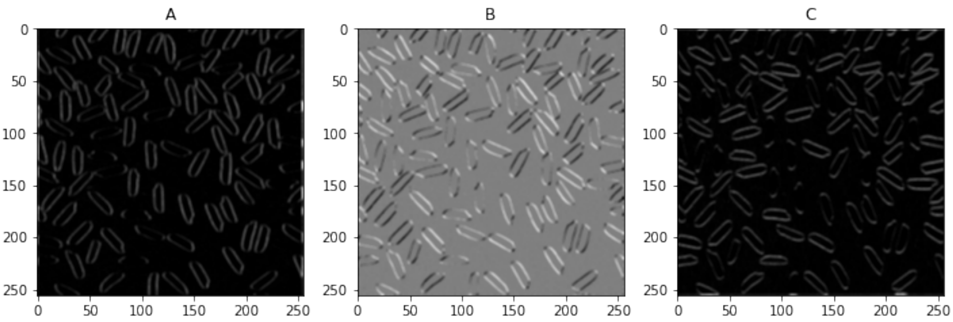
$$I_x(i,j) = \frac{I(i,j+1) - I(i,j-1)}{2}$$

$$I_y(i,j) = \frac{I(i-1,j) - I(i+1,j)}{2}$$

可以设计卷积核$G_x = \frac{1}{2}\begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}, G_y = \frac{1}{2}\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix}$分别计算 x 和 y 方向的梯度图（实际上我们可以选用长条形的 1×3、3×1 卷积，但方形的 3×3 卷积更容易理解）。将I转为 float32 数据类型后，使用 cv2.filter2D 函数完成卷积，设置 borderType 参数为 BORDER_CONSTANT 以对图像外圈补零，保证所有原始像素点均可求得梯度。结果如下：



3. 计算邻域内的梯度二次项。首先求出I_x^2, I_y^2, I_xI_y，然后构造卷积核$S = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$分别卷积三张二次项图来实现 3×3 邻域求和。进一步，使用 cv2.filter2D 函数完成卷积，同样对图像外圈补零，得到$A(i,j)$、$B(i,j)$、$C(i,j)$结果如下：



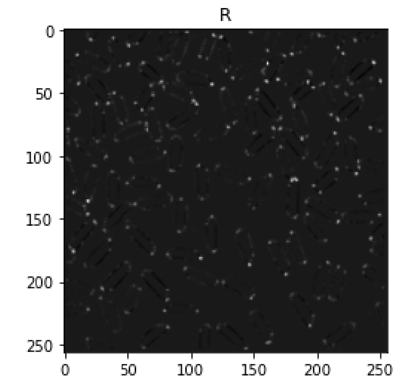
4. 用 A、B、C 的值的构成分块矩阵 M

$$M = \begin{bmatrix} A & B \\ B & C \end{bmatrix}$$

则可求取角点响应（corner-ness）R 值

$$R = \mathrm{Det}(M) - k\big(\mathrm{Tr}(M)\big)^2 = (AC - B^2) - k \times (A + C)^2$$

代入各图，结果如下：

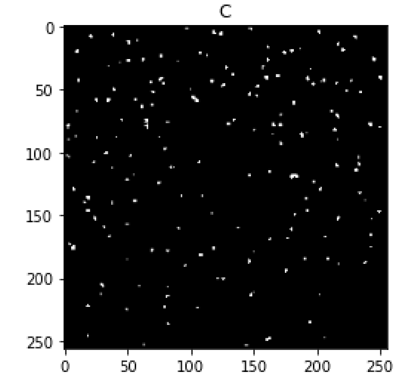


由于求取R值的参数是在 3×3 邻域操作后得到的，因此我们这里忽略了图像四周 2 像素宽的一圈（第一步求梯度图时引入 1 像素 padding，第三步邻域求和时引入 1 像素 padding，综合导致这里 2 像素宽的四周值无效），将其R置 0，因为那些数据不准确。

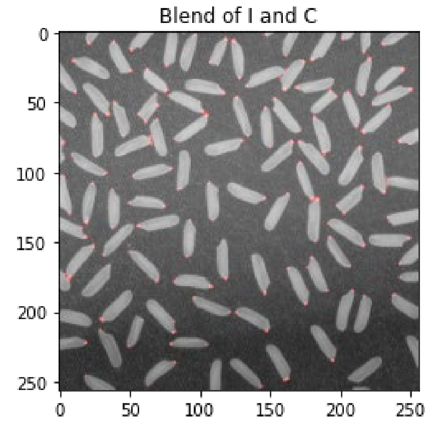
另外要注意，这里使用的是邻域求和后的图像参与运算，而不是求 Hessian 矩阵本身。

$$M \neq \begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix}$$

5. 统计得到$R_{\max}$。求得$R_{\max} = 41500812.0$，则阈值$0.2R_{\max} = 8300162.4$。将大于该阈值的点认为是角点，得到角点图C如下：

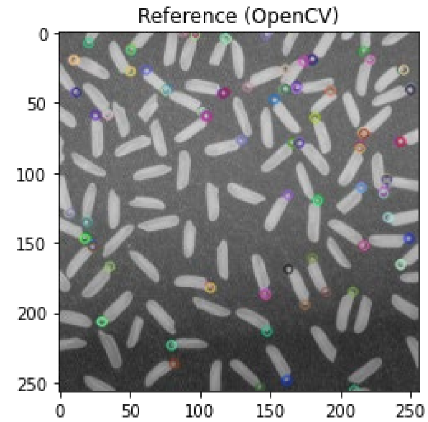


6. 将灰度图 I 转为 RGB 图，然后将 C 的幅度增强后叠加到 R 通道，即得到角点被红色标注出的结果图，如下：



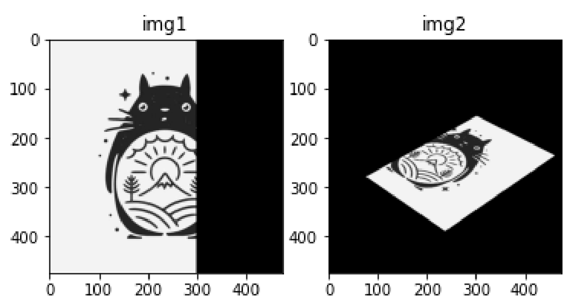
可以看到该算法确实检测出了图像中存在的“角点”位置。

后面我使用 OpenCV 的 Harris 角点检测器，使用相近的参数得到参考图像如下，两相对比，说明我们的结果正确。



Part 2 – 透视变换

1. 读取 img1 和 img2，利用之前实验的子程序即可完成，结果如下

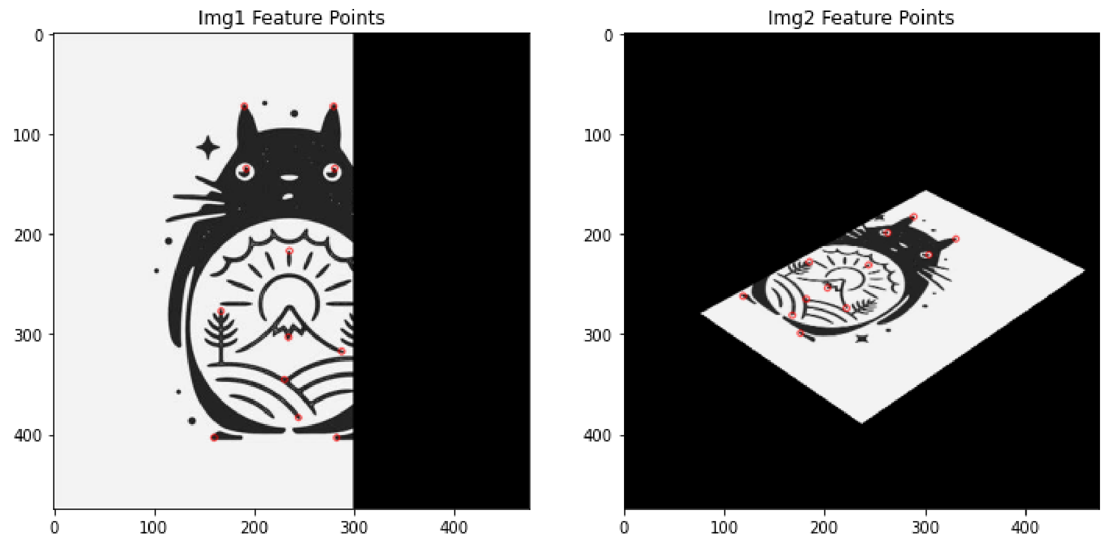


2. 我们试图确认一组 img1 到 img2 的变换，使得两图可以拼合出完整的原图。

3、4. 手工确认一系列 img1 与 img2 之间应该匹配的特征点，共 12 个，列表如下（标☆的是推荐点）：

序号	***img1*** **(*$x_{1,i}$*, *$y_{1,i}$*)**		***img2*** **(*$x_{2,i}$*, *$y_{2,i}$*)**	
	x **(row)**	***y*** **(col)**	***x*** **(row)**	***y*** **(col)**
1☆	73	190	183	288
2☆	73	279	205	330
3☆	135	192	199	262
4☆	135	280	221	303
5☆	277	167	228	184
6☆	317	287	274	221
7☆	403	160	262	118
8☆	403	282	299	175
9	217	235	231	243
10	303	234	254	202
11	383	244	281	167
12	345	230	265	181

将各点在图上用红圈标出后如下：



5. 接下来，目的是寻找变换

$$P\begin{bmatrix}x_2\\y_2\\1\end{bmatrix}=\begin{bmatrix}mx_1\\my_1\\m\end{bmatrix}，其中P=\begin{bmatrix}p_{11}&p_{12}&p_{13}\\p_{21}&p_{22}&p_{23}\\p_{31}&p_{32}&1\end{bmatrix}$$

按照作业讲义的说明，排布矩阵 A、向量 p 以及右手边，A 有 2×12 行 8 列：

```
A.shape
(24, 8)
```

，右手边是长为 24 的向量，然后得到方程组$A\vec{p}=\left[x_{1,1},y_{1,1},x_{1,2},y_{1,2},\dots,x_{1,n},y_{1,n}\right]^{\mathrm{T}}$，进而利用 A 的伪逆$(A^{\mathrm{T}}A)^{-1}A^{\mathrm{T}}$求解之，得到 p：

```
p = (np.linalg.inv(A.T @ A) @ A.T) @ rhs
p
```

```
array([ 1.78618194e+00, -1.07304042e+00,  6.24174078e+01,  1.80851482e+00,
        1.05527867e+00, -4.40601013e+02, -2.52605120e-05,  3.17646451e-05])
```

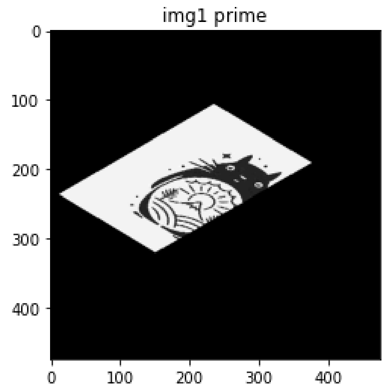
对应的矩阵 P 为：

```
P = np.array([*p, 1]).reshape(3, 3)
P
```

```
array([[ 1.78618194e+00, -1.07304042e+00,  6.24174078e+01],
       [ 1.80851482e+00,  1.05527867e+00, -4.40601013e+02],
       [-2.52605120e-05,  3.17646451e-05,  1.00000000e+00]])
```

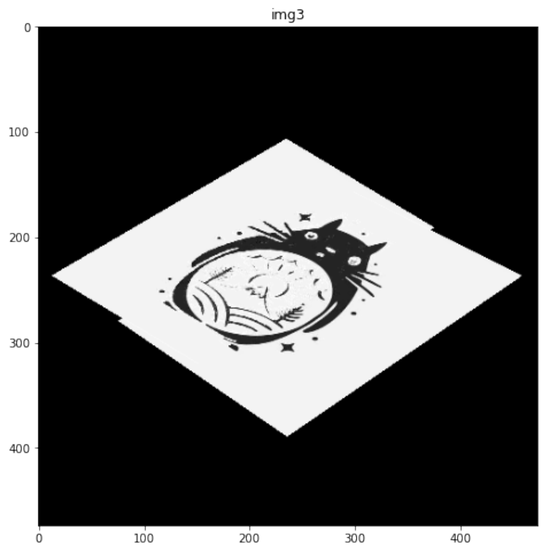
6、7. 我们遍历假设已知的变换后目标图像$img1'$中的所有坐标的齐次形式$(x_1', y_1', 1)$，在该向量左乘 P，即得到向量(mx_1, my_1, m)，伸缩将第三元化为 1（$x_1 = \frac{mx_1}{m}, y_1 = \frac{my_1}{m}$）即可求得对应原图像$img1$上的坐标，对此像素采样即得到$img1'$在该处的值。

为了完成这一步，对在亚像素上的点作快速最近邻插值（x、y 坐标直接四舍五入），超出原图像范围的像素点置 0，反复计算可得$img1'$如下：



可见该步骤完成了 img1 向 img2 的几何的透视变换过程。

8. 我们认为$img1'$足够准确，此时只需要与 img2 对各像素点对点计算最大值，即可得到融合后的图像 img3，结果如下：



整套流程基本完成了 img1 与 img2 的全图拼合过程，但仍存在一些误差，应该可以通过更多的特征点选择来优化。