OpenCV可以检测图像的主要特征，然后提取这些特征、使其成为图像描述符，这类似于人的眼睛和大脑。这些图像特征可作为图像搜索的数据库。此外，人们可以利用这些关键点将图像拼接起来，组成一个更大的图像，比如将许多图像放在一块，然后形成一个360度全景图像。

这里我们将学习使用OpenCV来检测图像特征，并利用这些特征进行图像匹配和搜索。我们会选取一些图像，并通过单应性，检测这些图像是否在另一张图像中。

[回到顶部](https://www.cnblogs.com/zyly/p/9508131.html#_labelTop)

**一 特征检测算法**

有许多用于特征检测和提取的算法，我们将会对其中大部分进行介绍。OpenCV最常使用的特征检测和提取算法有：

* Harris：该算法用于检测角点；
* SIFT：该算法用于检测斑点；
* SURF：该算法用于检测角点；
* FAST：该算法用于检测角点；
* BRIEF：该算法用于检测斑点；
* ORB：该算法代表带方向的FAST算法与具有旋转不变性的BRIEF算法；

通过以下方法进行特征匹配：

* 暴力(Brute-Force)匹配法；
* 基于FLANN匹配法；

可以采用单应性进行空间验证。

[回到顶部](https://www.cnblogs.com/zyly/p/9508131.html#_labelTop)

**二 特征定义**

那么，究竟什么是特征呢？为什么一副图像的某个特定区域可以作为一个特征，而其他区域不能呢？粗略的讲，特征就是有意义的图像区域，该区域具有独特特征和易于识别性。因此角点及高密度区域都是很好的特征，而大量重复的模式或低密度区域(例如图像中的蓝色天空)则不是很好的特征。边缘可以将图像分为两个区域，因此也可以看做好的特征。斑点是与周围有很大差别的像素区域，也是有意义的特征。

大多数特征检测算法都会涉及图像的角点、边和斑点的识别，也有一些涉及脊向的概念，可以认为脊向是细长物体的对称轴，例如识别图像中的一条路。角点和边都好理解，那什么是斑点呢？斑点通常是指与周围有着颜色和灰度差别的区域。在实际地图中，往往存在着大量这样的斑点，如一颗树是一个斑点，一块草地是一个斑点，一栋房子也可以是一个斑点。由于斑点代表的是一个区域，相比单纯的角点，它的稳定性要好，抗噪声能力要强，所以它在图像配准上扮演了很重要的角色。

由于某些算法在识别和提取某类型特征的时候有较好的效果，所以知道输入图像是什么很重要，这样做有利于选择最合适的OpenCV工具包。

[回到顶部](https://www.cnblogs.com/zyly/p/9508131.html#_labelTop)

**三 Harris检测角点特征**

在这之前其实我们已经接触过角点检测了，在相机标定的时候，我们就利用到了角点检测。不过那时候没有深入的去研究。在这里，我们将会深入原理去学习角点检测。

下面我们从使用cornerHarris()函数讲起。

cornerHarris(src, blockSize, ksize,k[,dst[,borderType]]);

参数详解:

* image:输入的单通道8位或者浮点图像；
* blockSize：就是扫描时候窗口的大小。
* ksize：cornerHarris函数会使用Sobel算子，该参数定义了Sobel算子的中孔。简单来说，该函数定义了角点检测的敏感度，其值必须介于3~31之间的奇数。
* k：harris 计算响应公式中的$k$值，一般取0.04~0.06；
* borderType：像素插值方法；

函数 cornerHarris 对输入图像进行 Harris 边界检测。输出是一幅浮点值图像，大小与输入图像大小相同，浮点值越高，表明越可能是特征角点(我们可以对图像进行阈值化)。

[复制代码](javascript:void(0);)

# -\*- coding: utf-8 -\*-

"""

Created on Mon Aug 20 20:17:34 2018

@author: lenovo

"""

'''

Harris角点检测

'''

import cv2

import numpy as np

img = cv2.imread('./image/cali.bmp')

img = cv2.resize(img,dsize=(600,400))

gray = cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY)

gray = np.float32(gray)

#角点检测 第三个参数为角点检测的敏感度，其值必须介于3~31之间的奇数

dst = cv2.cornerHarris(gray,3,23,0.04)

print(dst.shape) #(400, 600)

img[dst>0.01\*dst.max()] = [0,0,255]

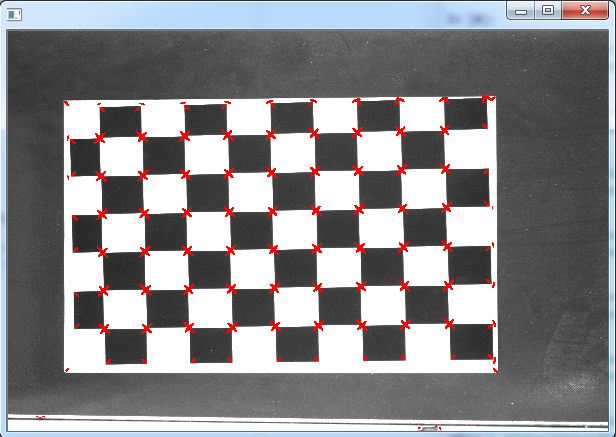
cv2.imshow('',img)

cv2.waitKey(0)

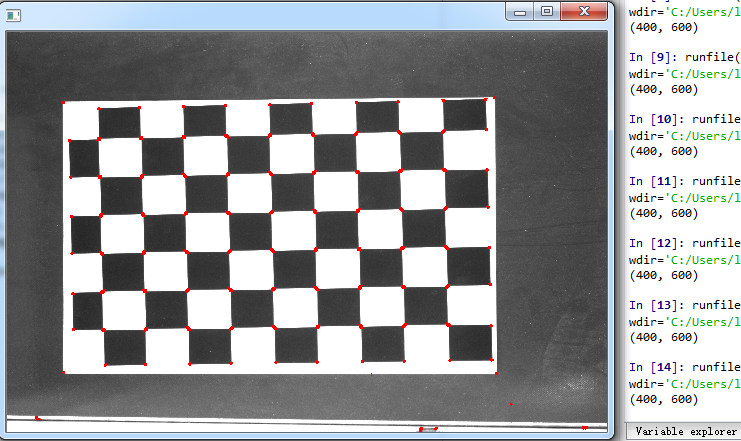
cv2.destroyAllWindows()

[复制代码](javascript:void(0);)

运行结果如下：



如果我们把第三个参数改为3:，可以看到：



[回到顶部](https://www.cnblogs.com/zyly/p/9508131.html#_labelTop)

**四 Harris检测原理**

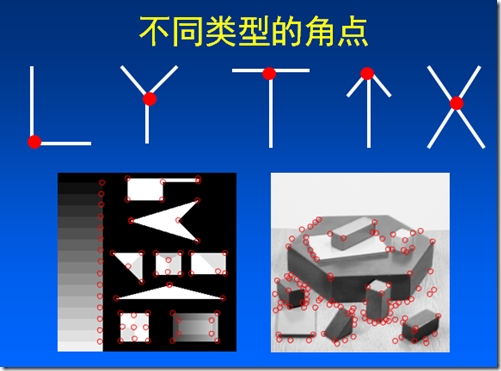
上面我们已经通过实例演示了Harris检测的效果，相信你对Harris角点检测已经有了初步的认识。这里我将带你深入了解Harris角点检测的原理。

我们先来看一幅图片，了解一下什么是角点？



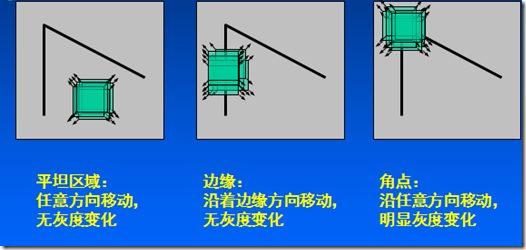
上图中E,F中的角我们通常称作角点（corner points），他们具有以下特征：

* 轮廓之间的交点；
* 对于同一场景，即使视角发生变化，通常具备稳定性质的特征；
* 该点附近区域的像素点无论在梯度方向上还是其梯度幅值上有着较大变化；



harris特征角最早在paper A Combined Corner and Edge Detector中被Chris Harris & Mike Stephens提出。

Harris角点检测的基本思想：算法基本思想是使用一个固定窗口在图像上进行任意方向上的滑动，比较滑动前与滑动后两种情况，窗口中的像素灰度变化程度，如果存在任意方向上的滑动，都有着较大灰度变化，那么我们可以认为该窗口中存在角点。



**4.1 灰度变化描述**

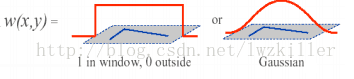
当窗口发生$[u,v]$移动时，那么滑动前与滑动后对应的窗口中的像素点灰度变化描述如下：

$$E(u,v)=\sum\limits\_{(x,y)€W}w(x,y)[I(x+u,y+v)-I(x,y)]^2$$

参数解释：

* $[u,v]$是窗口$W$的偏移量；
* $(x,y)$是窗口$W$所对应的像素坐标位置，窗口有多大，就有多少个位置；
* $I(x,y)$是像素坐标位置$(x,y)$的图像灰度值；
* $I(x+u,y+v)$是像素坐标位置$(x+u,y+v)$的图像灰度值；
* $w(x,y)$是窗口函数，最简单情形就是窗口$W$内的所有像素所对应的$w$权重系数均为1.但有时候，我们会将$w(x,y)$函数设置为以窗口$W$中心为原点的二元正太分布。如果窗口$W$中心点是角点时，移动前与移动后，该点在灰度变化贡献最大；而离窗口$W$中心(角点)较远的点，这些点的灰度变化几近平缓，这些点的权重系数，可以设定小值，以示该点对灰度变化贡献较小，那么我们自然而然想到使用二元高斯函数来表示窗口函数；

我们的窗口函数通常有如下两种形式：



根据上述表达式，当窗口在平潭区域上移动，可以想象得到，灰度不会发生什么变换。$E(u,v)=0$；如果窗口处在纹理比较丰富的区域上滑动，那么灰度变化会很大。算法最终思想就是计算灰度发生较大变化时所对应的位置，当然这个较大是指任意方向上的滑动，并非单指某个方向。

**4.2 $E(u,v)$化简**

首先需要了解泰勒公式，任何一个函数表达式，均可有泰勒公式进行展开，以逼近原函数，我们可以对下面函数进行一阶展开(如果对泰勒公式忘记了，可以翻翻本科所学的高等数学)。

$$f(x+u,y+v)≈f(x,y)+uf\_x(x,y)+vf\_y(x,y)$$

那么

$$\sum\limits\_{(x,y)€W}w(x,y)[I(x+u,y+v)-I(x,y)]^2$$

$$≈\sum\limits\_{(x,y)€W}w(x,y)[I(x,y)+uI\_x+vI\_y-I(x,y)]^2$$

$$=\sum\limits\_{(x,y)€W}w(x,y)[u^2I\_x^2+2uvI\_xI\_y+v^2I\_y^2]$$

$$=\sum\limits\_{(x,y)€W}w(x,y)\begin{bmatrix}u & v\end{bmatrix}\begin{bmatrix} I\_x^2 & I\_xIy \\   I\_xI\_y & I\_y^2\end{bmatrix}\begin{bmatrix}u \\ v\end{bmatrix}$$

$$=\begin{bmatrix}u & v\end{bmatrix}(\sum\limits\_{(x,y)€W}w(x,y)\begin{bmatrix} I\_x^2 & I\_xIy \\   I\_xI\_y & I\_y^2\end{bmatrix})\begin{bmatrix}u \\ v\end{bmatrix}$$

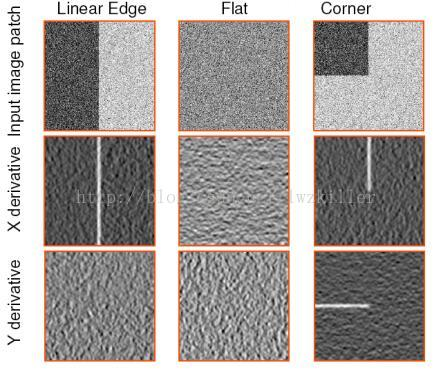
所以$E(u,v)$表达式可以更新为：

$$E(u,v)=\begin{bmatrix}u & v\end{bmatrix}M\begin{bmatrix}u \\ v\end{bmatrix}$$

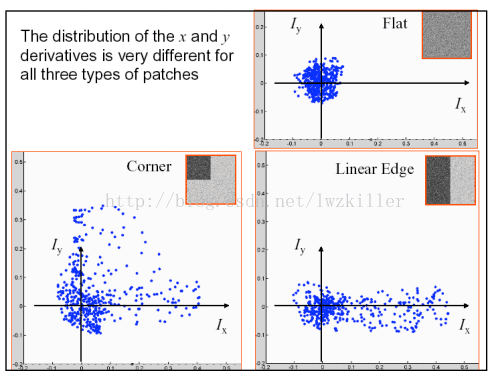
其中：$M=\sum\limits\_{(x,y)€W}w(x,y)\begin{bmatrix} I\_x^2 & I\_xIy \\   I\_xI\_y & I\_y^2\end{bmatrix}$，$I\_x$,$I\_y$分别为窗口内像素点$(x,y)$在$x$方向上和$y$方向上的梯度值。

**4.3 矩阵$M$的关键性**

难道我们是直接求上述的$E(u,v)$值来判断角点吗？Harris角点检测并没有这样做，而是通过对窗口内的每个像素的$x$方向上的梯度与$y$方向上的梯度进行统计分析。这里以$I\_x$和$I\_y$为坐标轴，因此每个像素的梯度坐标可以表示成$(I\_x,I\_y)$。针对平坦区域，边缘区域以及角点区域三种情形进行分析：



下图是对这三种情况窗口中的对应像素的梯度分布进行绘制：



不知道大家有没有注意到这三种区域的特点:

* 平坦区域上的每个像素点所对应的$(I\_x,I\_y)$坐标分布在原点附近，其实也很好理解，针对平坦区域的像素点，他们的梯度方向虽然各异，但是其幅值都不是很大，所以均聚集在原点附近；
* 边缘区域有一坐标轴分布较散，至于是哪一个坐标上的数据分布较散不能一概而论，这要视边缘在图像上的具体位置而定，如果边缘是水平或者垂直方向，那么$I\_y$轴方向或者$I\_x$方向上的数据分布就比较散；
* 角点区域的$x$、$y$方向上的梯度分布都比较散。

我们是不是可以根据这些特征来判断哪些区域存在角点呢？

上面我们已经计算出了$M$矩阵:

$M=\sum\limits\_{(x,y)€W}w(x,y)\begin{bmatrix} I\_x^2 & I\_xIy \\   I\_xI\_y & I\_y^2\end{bmatrix}=\begin{bmatrix} A & C \\   C & B \end{bmatrix}$

我们可以将$E(u,v)$近似为二项函数：

$$E(u,v)=Au^2+2Cuv+Bv^2$$

其中

$$A=\sum\limits\_{(x,y)€W}w(x,y)\*I\_x^2$$

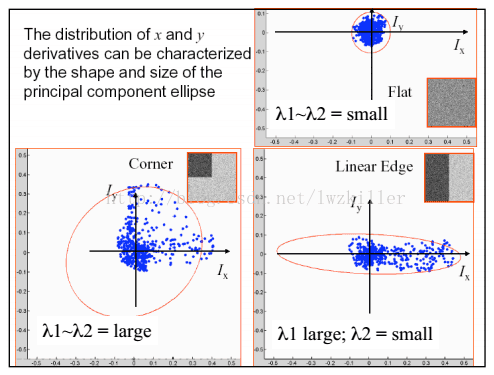
$$B=\sum\limits\_{(x,y)€W}w(x,y)\*I\_y^2$$

$$C=\sum\limits\_{(x,y)€W}w(x,y)\*I\_xI\_y$$

二次项函数本质上就是一个椭圆函数。椭圆的长和宽是由$M$的特征值$λ\_1,λ\_2$决定的(椭圆的长短轴正是矩阵$M$特征值平方根的倒数)，椭圆的方向是由$M$的特征向量决定的，椭圆方程为：

$$\begin{bmatrix} u & v \end{bmatrix}M\begin{bmatrix} u \\ v \end{bmatrix}=1$$

如果使用椭圆进行数据集表示，则绘制图示如下：



虽然我们利用$E(u,v)$来描述角点的基本思想，然而最终我们仅仅使用的是矩阵$M$。让我们看看矩阵$M$形式，是不是跟协方差矩阵形式很像，像归像，但是还是有些不同，哪儿不同？一般协方差矩阵对应维的随机变量需要减去该维随机变量的均值：

把$I\_x$,$I\_y$看成两个字段，假设窗口内有m个像素点，也就是等价于有m个样本，我们先计算每个字段的均值：

$$\bar{I\_x}=\sum\limits\_{i=1}^{m}I\_{xi}$$

$$\bar{I\_y}=\sum\limits\_{i=1}^{m}I\_{yi}$$

我们仍然使用$(I\_{xi},I\_{yi})$表示样本$(I\_{xi},I\_{yi})$去均值后的值，则由这m个样本组成的矩阵：

$$X=\begin{bmatrix} I\_{x1} & I\_{x2} & ... & I\_{xm} \\ I\_{y1} & I\_{y2} & ... & I\_{ym}\end{bmatrix}$$

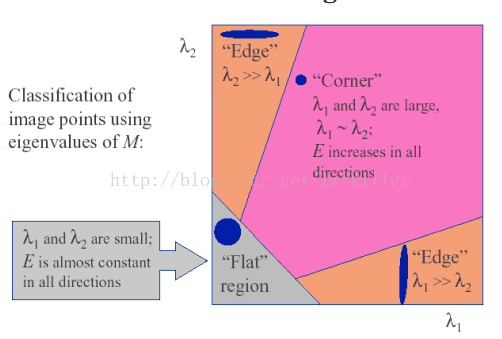
则对应协方差矩阵可以写成：$$C=\frac{1}{m}XX^T=\frac{1}{m}\sum\limits\_{i=1}^{m}\begin{bmatrix} I\_x^2 & I\_xIy \\ I\_xI\_y & I\_y^2 \end{bmatrix}$$

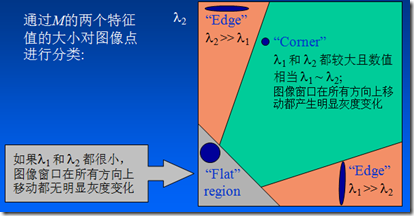
但矩阵$M$中并没有这样做，所以在矩阵$M$里，我们先进行各维的均值化处理，那么各维所对应的随机变量的均值为0，协方差矩阵就大大简化了，简化的最终结果就是矩阵$M$，是否明白了(注意为了简化运算，我们先假设$M$矩阵中的权重系数$w(x,y)=1$，并且省略掉了求均值)我们的目的是分析数据的主要成分，相信了解PCA原理的，应该都了解均值化的作用。$$M=\sum\limits\_{(x,y)€W}\begin{bmatrix} I\_x^2 & I\_xIy \\   I\_xI\_y & I\_y^2\end{bmatrix}$$

如果我们对协方差矩阵$M$进行对角化，很明显，其对角线就是各个字段的方差，这点大家应该明白吧？不明白的话可以复习下PCA原理。

* 如果两个字段$(I\_x,I\_y)$所对应的特征值都比较大，说明什么？ 像素点的梯度分布比较散，梯度变化程度比较大，符合角点在窗口区域的特点；
* 如果是平坦区域，那么像素点的梯度所构成的点集比较集中在原点附近，因为窗口区域内的像素点的梯度幅值非常小，此时矩阵$M$的对角化的两个特征值比较小；
* 如果是边缘区域，在计算像素点的$x$、$y$方向上的梯度时，边缘上的像素点的某个方向的梯度幅值变化比较明显，另一个方向上的梯度幅值变化较弱，其余部分的点都还是集中原点附近，这样$M$对角化后的两个特征值理论应该是一个比较大，一个比较小，当然对于边缘这种情况，可能是呈45°的边缘，致使计算出的特征值并不是都特别的大，总之跟含有角点的窗口的分布情况还是不同的。

注：$M$为协方差矩阵，需要大家自己去理解下，窗口中的像素集构成一个矩阵（$X€R^{2×m}$，假设这里有m个像素点），使用该矩阵乘以该矩阵的转置，即是协方差矩阵。





因此可以得出下列结论：

* 特征值都比较大时，即窗口中含有角点；
* 特征值一个较大，一个较小，窗口中含有边缘；
* 特征值都比较小，窗口处在平坦区域；

**4.4 如何度量角点响应**

通常用下面表达式进行度量，对每一个窗口计算得到一个分数$R$，根据$R$的大小来判定窗口内是否存在harris特征角。分数$R$根据下面公式计算得到：

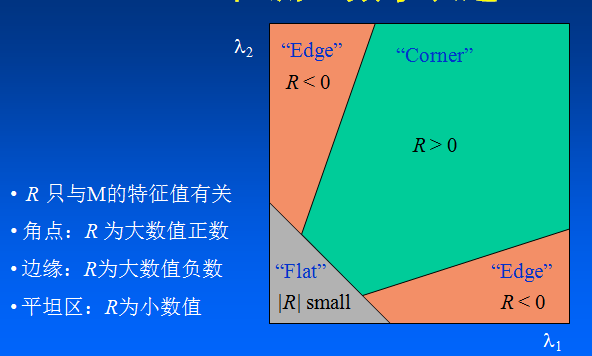
$$R=det(M)-k(trace(M))^2$$

$$det(M)=λ\_1λ\_2$$

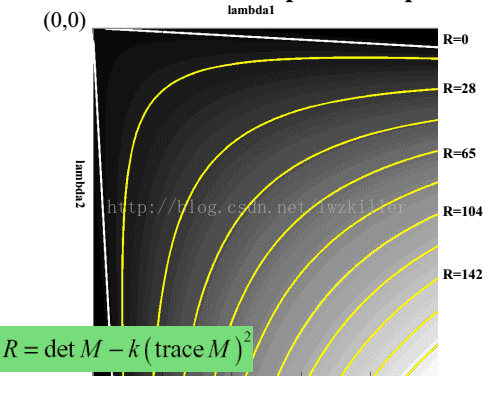
$$trace(M)=λ\_1+λ\_2$$

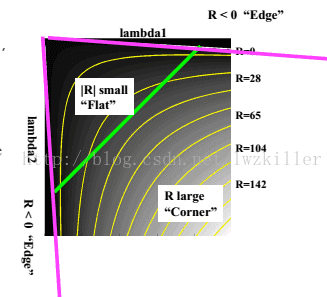
这里$λ\_1,λ\_2$是矩阵$M$的2个特征值，$k$是一个指定值，这是一个经验参数，需要实验确定它的合适大小，通常它的值在0.04和0.06之间,它的存在只是调节函数的形状而已。

 $R$取决于$M$的特征值，对于角点$|R|$很大，平坦的区域$|R|$很小，边缘的$R$为负值;



但是为什么会使用这样的表达式呢？一下子是不是感觉很难理解？其实也不难理解，函数表达式一旦出来，我们就可以绘制它的图像，而这个函数图形正好满足上面几个区域的特征。 通过绘制函数图像，直观上更能理解。绘制的$R$函数图像如下：





所以说难点不在于理解这个函数表达式，而在于如何创造出这个函数表达式。Harris也许对很多函数模型非常了解，对于创造出这样的一个函数表达式，易如反掌，当然在我们看来感觉是很了不起的，那是因为我们见过的函数模型太少。

最后设定R的阈值，进行角点判断。当然其中还有些后处理步骤就不再说了，比如说角点的极大值抑制等。

非极大值抑制的原理，在一个窗口内，如果有很多角点则用值最大的那个角点，其他的角点都删除。

注意：有些朋友在问是如何通过矩阵判断角点的？ 其实上面，我们已经推导出$E(u,v)$的表达式，大家看看这个表达式有什么特征，其中矩阵M是实对称矩阵，那么E表达式其实就是二次型，对于二次型想必大家会有印象，$u$,$v$代表窗口滑动方向以及滑动量，$E$代表灰度变化，通过矩阵M进行特征值求解，而特征值所对应的特征向量即为灰度变化方向。如果两个特征值较大，则表示有两个方向灰度变化较快。所以可以直接通过求解$M$的特征值进行角点判断.

[回到顶部](https://www.cnblogs.com/zyly/p/9508131.html#_labelTop)

**五 自己实现Harris角点检测**

**5.1 实现步骤**

Harris角点检测可以分为5个步骤：

1、计算图像$I(x,y)$在$x$和$y$两个方向的梯度$I\_x$,$I\_y$;

$$I\_x=\frac{∂I}{∂x}=I×\begin{bmatrix} -1 & 0 & 1 \\  -2 & 0 & 2 \\ -1 & 0 & -1  \end{bmatrix}$$

$$I\_y=\frac{∂I}{∂y}=I×\begin{bmatrix} -1 & -2 & -1 \\  0 & 0 & 0 \\ 1 & 2 & 1  \end{bmatrix}$$

其中$I$为3×3邻域矩阵。

2、计算图像两个方向梯度的乘积;

$$I\_x^2=I\_x\*I\_x$$

$$I\_y^2=I\_y\*I\_y$$

$$I\_xI\_y=I\_x\*I\_y$$

3、使用高斯函数对$I\_x^2$、$I\_y^2$、$I\_xI\_y$进行高斯加权(取σ=2,ksize=3)，计算中心点为$(x,y)$的窗口$W$对应的矩阵$M$；

$$A=\sum\limits\_{(x,y)€W}g(I\_x^2)=\sum\limits\_{(x,y)€W}I\_x^2\*w(x,y)$$

$$B=\sum\limits\_{(x,y)€W}g(I\_y^2)=\sum\limits\_{(x,y)€W}I\_y^2\*w(x,y)$$

$$C=\sum\limits\_{(x,y)€W}g(I\_xI\_y)=\sum\limits\_{(x,y)€W}I\_xI\_y\*w(x,y)$$

其中$M=\begin{bmatrix} A& C \\C  & B\end{bmatrix}=\sum\limits\_{(x,y)€W}w(x,y)\begin{bmatrix} I\_x^2 & I\_xIy \\   I\_xI\_y & I\_y^2\end{bmatrix}$.

4、计算每个像素点$(x,y)处的$Harris响应值$R$；

$$R=det(M)-k(trace(M))^2$$

5、过滤大于某一阈值$t$的$R$值；

$$R=\{R:det(M)-k(trace(M))^2 > t \}$$

如果需要在3×3或者5×5的邻域进行非最大值抑制，则局部最大值点即为图像中的角点。

以上所有公式\*表示逐像素点乘。

**5.2 实现代码**

[复制代码](javascript:void(0);)

# -\*- coding: utf-8 -\*-

"""

Created on Mon Aug 20 20:17:34 2018

@author: lenovo

"""

'''

Harris角点检测

'''

import cv2

import numpy as np

def test():

'''

调用系统库函数进行测试

'''

img = cv2.imread('./image/cali.bmp')

img = cv2.resize(img,dsize=(600,400))

gray = cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY)

gray = np.float32(gray)

#角点检测 第三个参数为角点检测的敏感度，其值必须介于3~31之间的奇数

dst = cv2.cornerHarris(gray,3,3,0.04)

print(dst.shape) #(400, 600)

img[dst>0.01\*dst.max()] = [0,0,255]

cv2.imshow('',img)

cv2.waitKey(0)

cv2.destroyAllWindows()

def harris\_detect(img,ksize=3):

'''

自己实现角点检测

params:

img:灰度图片

ksize：Sobel算子窗口大小

return：

corner：与源图像一样大小，角点处像素值设置为255

'''

k = 0.04 #响应函数k

threshold = 0.01 #设定阈值

WITH\_NMS = False #是否非极大值抑制

#1、使用Sobel计算像素点x,y方向的梯度

h,w = img.shape[:2]

#Sobel函数求完导数后会有负值，还有会大于255的值。而原图像是uint8，即8位无符号数，所以Sobel建立的图像位数不够，会有截断。因此要使用16位有符号的数据类型，即cv2.CV\_16S。

grad = np.zeros((h,w,2),dtype=np.float32)

grad[:,:,0] = cv2.Sobel(img,cv2.CV\_16S,1,0,ksize=3)

grad[:,:,1] = cv2.Sobel(img,cv2.CV\_16S,0,1,ksize=3)

#2、计算Ix^2,Iy^2,Ix\*Iy

m = np.zeros((h,w,3),dtype=np.float32)

m[:,:,0] = grad[:,:,0]\*\*2

m[:,:,1] = grad[:,:,1]\*\*2

m[:,:,2] = grad[:,:,0]\*grad[:,:,1]

#3、利用高斯函数对Ix^2,Iy^2,Ix\*Iy进行滤波

m[:,:,0] = cv2.GaussianBlur(m[:,:,0],ksize=(ksize,ksize),sigmaX=2)

m[:,:,1] = cv2.GaussianBlur(m[:,:,1],ksize=(ksize,ksize),sigmaX=2)

m[:,:,2] = cv2.GaussianBlur(m[:,:,2],ksize=(ksize,ksize),sigmaX=2)

m = [np.array([[m[i,j,0],m[i,j,2]],[m[i,j,2],m[i,j,1]]]) for i in range(h) for j in range(w)]

#4、计算局部特征结果矩阵M的特征值和响应函数R(i,j)=det(M)-k(trace(M))^2 0.04<=k<=0.06

D,T = list(map(np.linalg.det,m)),list(map(np.trace,m))

R = np.array([d-k\*t\*\*2 for d,t in zip(D,T)])

#5、将计算出响应函数的值R进行非极大值抑制，滤除一些不是角点的点，同时要满足大于设定的阈值

#获取最大的R值

R\_max = np.max(R)

#print(R\_max)

#print(np.min(R))

R = R.reshape(h,w)

corner = np.zeros\_like(R,dtype=np.uint8)

for i in range(h):

for j in range(w):

if WITH\_NMS:

#除了进行进行阈值检测 还对3x3邻域内非极大值进行抑制(导致角点很小，会看不清)

if R[i,j] > R\_max\*threshold and R[i,j] == np.max(R[max(0,i-1):min(i+2,h-1),max(0,j-1):min(j+2,w-1)]):

corner[i,j] = 255

else:

#只进行阈值检测

if R[i,j] > R\_max\*threshold :

corner[i,j] = 255

return corner

if \_\_name\_\_=='\_\_main\_\_':

img = cv2.imread('./image/cali.bmp')

img = cv2.resize(img,dsize=(600,400))

#转换为灰度图像

gray = cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY)

dst = harris\_detect(gray)

print(dst.shape) #(400, 600)

img[dst>0.01\*dst.max()] = [0,0,255]

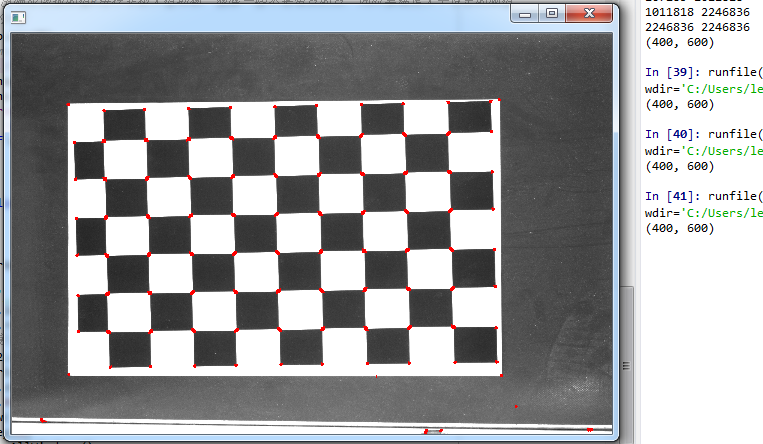
cv2.imshow('',img)

cv2.waitKey(0)

cv2.destroyAllWindows()

[复制代码](javascript:void(0);)

运行效果如下：



[回到顶部](https://www.cnblogs.com/zyly/p/9508131.html#_labelTop)

**六 Harries角点的性质**

**6.1 参数$k$对角点检测的影响**

假设已经得到了矩阵$M$的特征值$λ\_1≥λ\_2≥0$，令$λ\_2=αλ\_1$，$0≤α≤1$。由特征值与矩阵$M$的直迹和行列式的关系可得：

$$detM=\prod\limits\_{i}λ\_i$$  $$trace(M)=\sum\limits\_{i}λ\_i$$

从而可以得到角点的响应：

$$R=λ\_1λ\_2-k(λ\_1+λ\_2)^2=λ\_1^2(α-k(1+α)^2)$$

假设$R≥0$，则有：

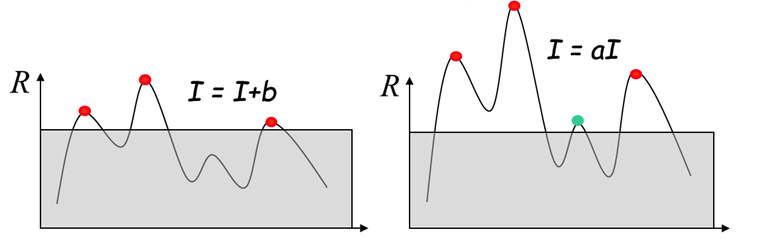
$$0≤k≤\frac{α}{(1+α)^2}≤0.25$$

对于较小的$α$值，$R≈λ\_1^2(α-k),α<k$。

由此，可以得出这样的结论，增大$k$的值，降低角点检测的灵敏度，减少被检测角点的数量；减少$k$值，增加角点检测的灵敏度，增加被检测角点的数量。

**6.2 Harris角点检测算子对亮度和对比度的变化不灵敏**

这是因为在进行Harris角点检测时，使用了微分算子对图像进行微分运算，而微分运算对图像密度的拉升或收缩和对亮度的抬高或下降不敏感。换言之，对亮度和对比度的仿射变换并不改变Harris响应的极值点出现的位置，但是，由于阈值的选择，可能会影响角点检测的数量。



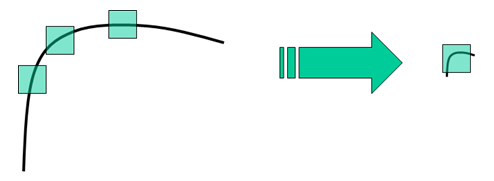
左图表示亮度变化，右图表示对比度变化。

**6.3 Harris角点检测算子具有旋转不变性**

Harris角点检测算子使用的是角点附近的区域灰度二阶矩矩阵。而二阶矩矩阵可以表示成一个椭圆，椭圆的长短轴正是二阶矩矩阵特征值平方根的倒数。当特征椭圆转动时，特征值并不发生变化，所以判断角点响应值$R$也不发生变化，由此说明Harris角点检测算子具有旋转不变性。

**6.4 Harris角点检测算子不具有尺度不变性**

如下图所示，当图像被缩小时，在检测窗口尺寸不变的前提下，在窗口内所包含图像的内容是完全不同的。左侧的图像可能被检测为边缘或曲线，而右侧的图像则可能被检测为一个角点。



[回到顶部](https://www.cnblogs.com/zyly/p/9508131.html#_labelTop)

**七 代码下载**

[Young](https://gitee.com/zyly2033) / [opencv](https://gitee.com/zyly2033/opencv)

**参考文献：**

[**[1]OpenCV教程(43) harris角的检测(1)**](https://www.cnblogs.com/mikewolf2002/p/3547436.html)

[**[2]Harris角点检测原理详解**](https://blog.csdn.net/lwzkiller/article/details/54633670)

[**[3]PCA的数学原理**](http://blog.codinglabs.org/articles/pca-tutorial.html)

[**[4]图像处理基础(4)：高斯滤波器详解**](https://www.cnblogs.com/wangguchangqing/p/6407717.html)

[**[5]Harris角点**](https://www.cnblogs.com/ronny/p/4009425.html)

[**[6]『Python』Harris角点检测代码实现**](https://www.cnblogs.com/hellcat/p/8806306.html)

[**[7]斑点检测**](https://www.cnblogs.com/ronny/p/3895883.html)

[**[8]Harris原理及源码解析(C++)**](http://blog.csdn.net/luoshixian099/article/details/48244255)