# 一、什么是sift特征？如何计算？

1. sift = cv2.xfeatures2d.SIFT\_create() 实例化

参数说明：sift为实例化的sift函数

2. kp = sift.detect(gray, None)  找出图像中的关键点

参数说明: kp表示生成的关键点，gray表示输入的灰度图，

3. ret = cv2.drawKeypoints(gray, kp, img) 在图中画出关键点

参数说明：gray表示输入图片, kp表示关键点，img表示输出的图片

4.kp, dst = sift.compute(kp) 计算关键点对应的sift特征向量

参数说明：kp表示输入的关键点，dst表示输出的sift特征向量，通常是128维的

第一步：进行高斯模糊，获得不同模糊度的图片

第二步：进行直接的降采样，获得多分辨的图片

第三步：将高斯模糊后的图片进行相减操作获得差分金字塔，使用DOC公式

第四步：对差分金字塔进行上下金字塔的比较，获得极值点

第五步：使用泰勒展开式，获得精确的极值点

第六步：使用herrian公式，通过特征向量变化，比较λ1和λ2的大小，用于消除边界点

第七步：使用sobel算子，计算特征点的梯度大小和梯度方向

第八步：统计特征点临近位置的梯度方向，做出直方图，求出特征点的主要方向

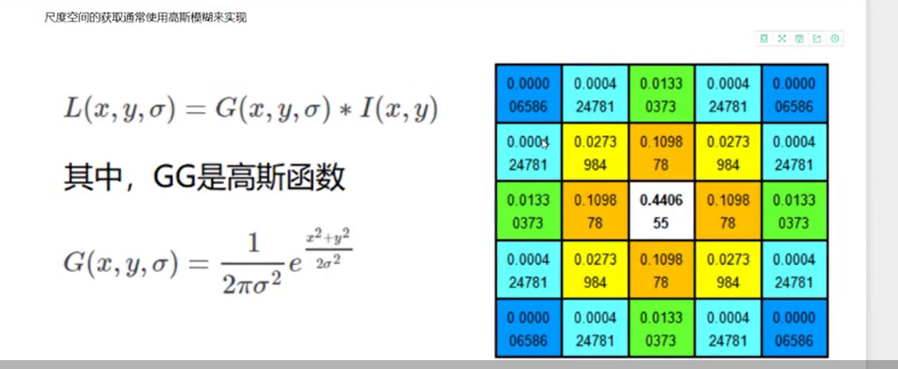
第九步：对应于特征的主要方向，进行旋转，保持特征点大小的方向不变性

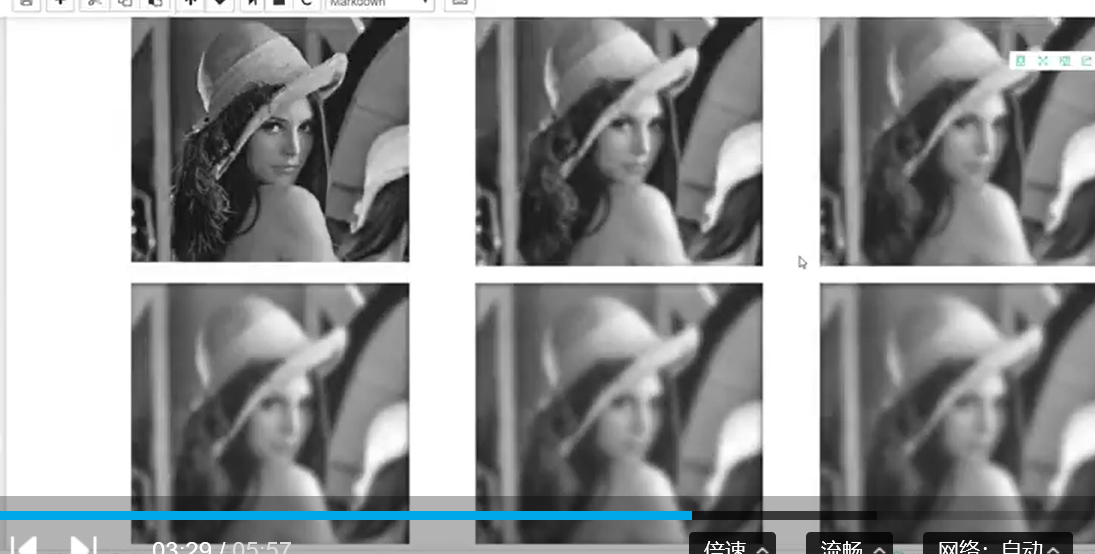
第十步：对领域的特征点进行统计，按照4\*4的数目，每个区域生成8个方向，即每个方向的出现的次数作为特征，一般使用16个区域，即16\*8= 128个特征

shift特征点：用来进行侦查和描述图片的特征，它在空间尺度中寻找极值点，并提取位置，尺度(梯度大小)，旋转不变量(方向)

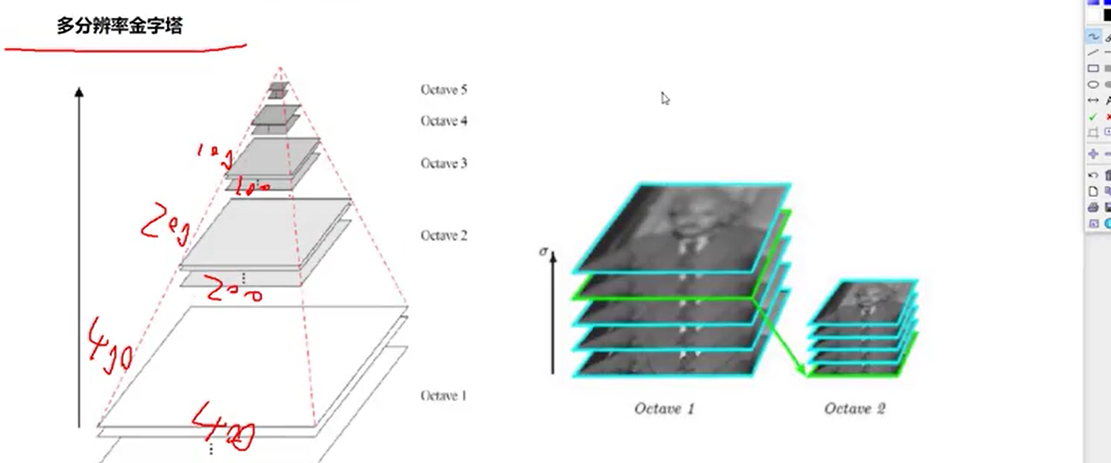
流程：

第一步：先对图像进行高斯模糊，获得5-6张的模糊后的图片，高斯方程为G(x, y, σ) = 1/(2pi\*σ^2) \* e^-(x^2+y^2)/2σ^2，σ越大，高斯模糊的程度就越高

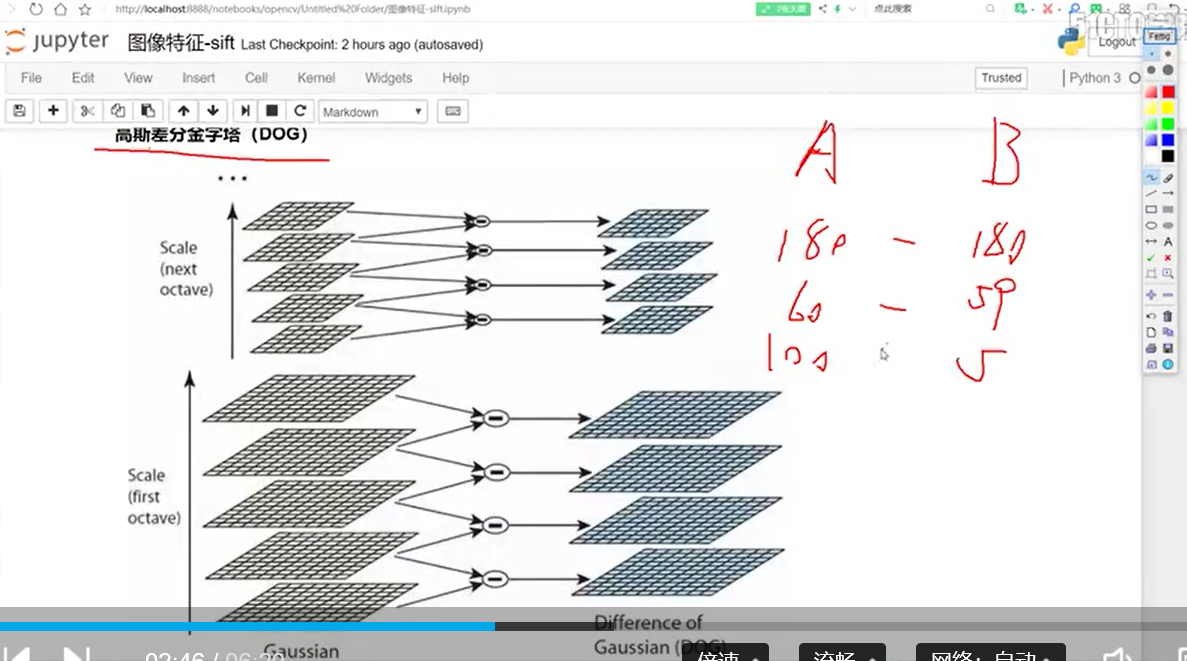
 这里的G(x, y, σ)方程存在一些错误

这是高斯模糊后的图片

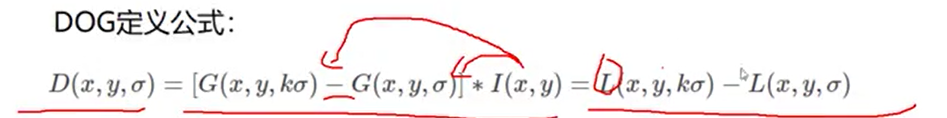
第二步：构造多分辨金字塔，多分辨金字塔的构造直接使用降采样不需要模糊的操作，这里可以使用平均降采样



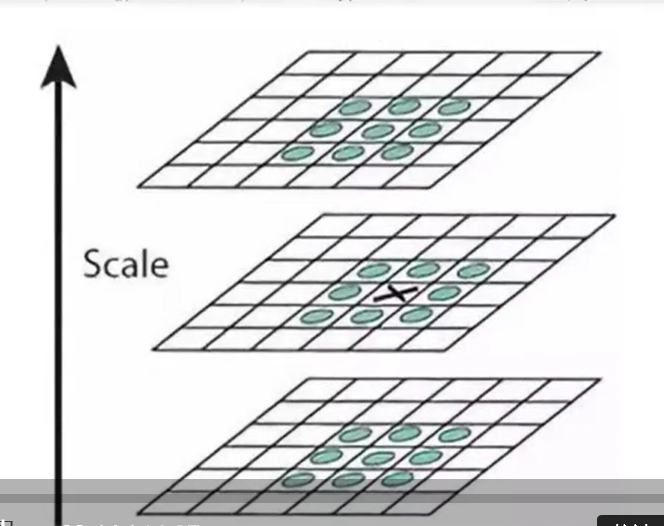
第三步：构造高斯差分金字塔， 图中的每组5张图片为原始图片经过不同σ高斯参数模糊后获得的图。将5张图进行上下的相减操作，获得右边的差分图



下面这个式子表示的是高斯差分金字塔，即不同的高斯项进行相减，最后\*I(x, y)表示差分金字塔的大小值

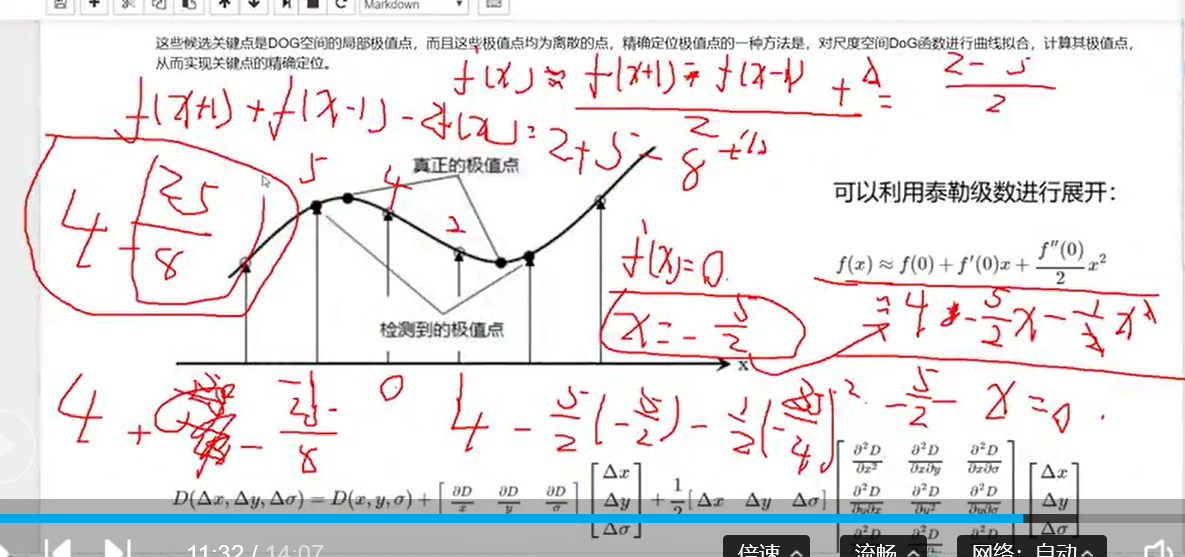


第四步：对获得的高斯差分金字塔，查找极值点， 对于一个点是否是极值点，将其上面一幅图对应的9个点+下面一幅图对应的9个点，加上该点周围的8的点，判断这个点是否是极值点

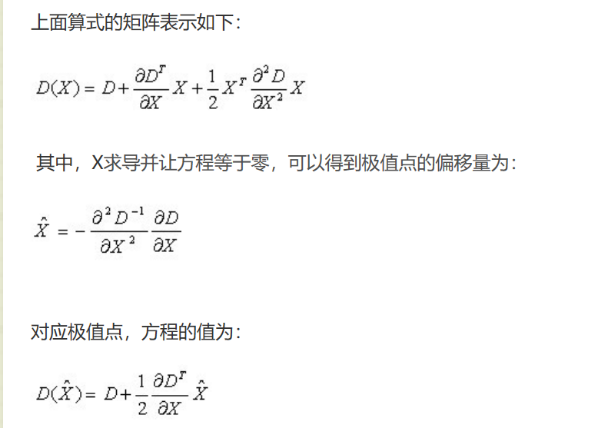


第五步：如果是极值点，即为关键点，这里我们对关键点做一个精确的定位，这里使用泰勒公式进行展开

D(x) = D + ∂ D^T / ∂x \* x + 1/2 \* x^T \* ∂D^2 / ∂^X^2 \* x  x表示的是x轴上的偏移量，对x进行求导等于0，解得最终的结果代回D(x),D(x)为最终的极值点

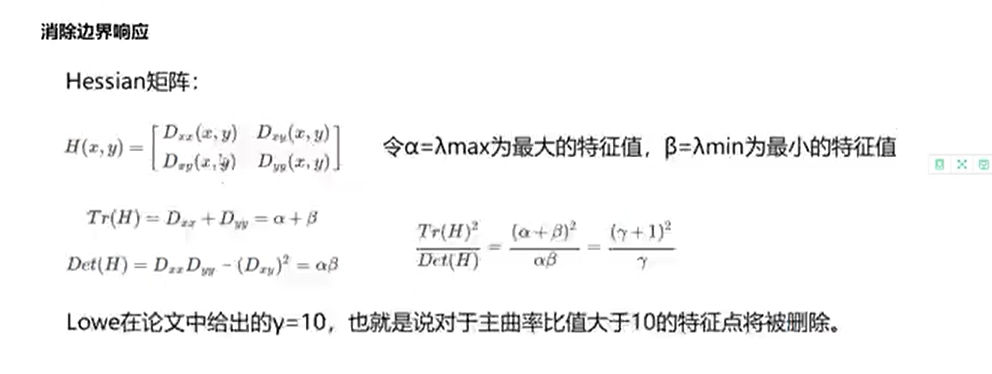
该图使用简化的泰勒展开式，求解0这一点的近似值

最下面的D(x, y, z) 为泰勒的二阶展开式

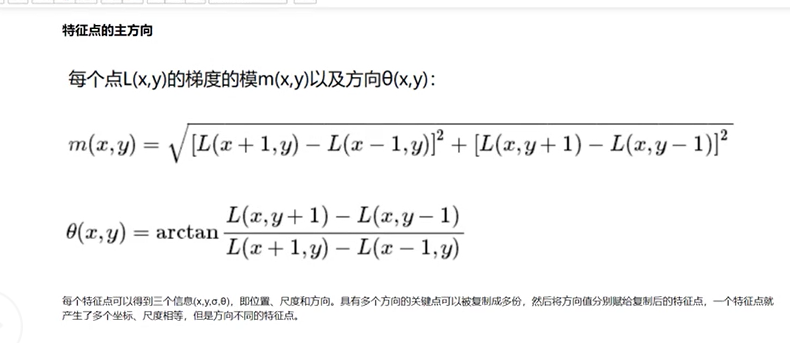
将上述进行简单表示，这就是偏移的D(x)， 即获得实际的极值点

第六步：消除边界效应

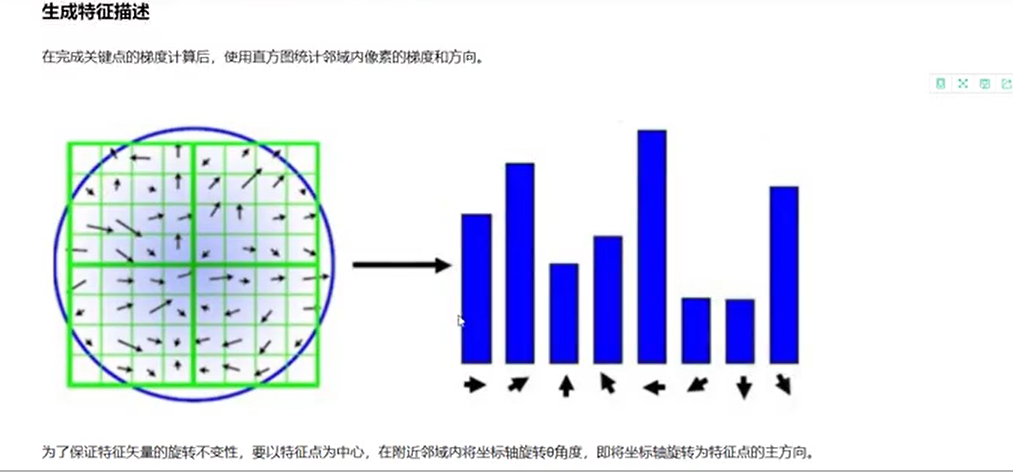
使用harris角点检测的原理， 求出H(x, y) 即构造的梯度变化矩阵，求解λ1和λ2, 如果λ1>>λ2则表示为边界点，进行去除



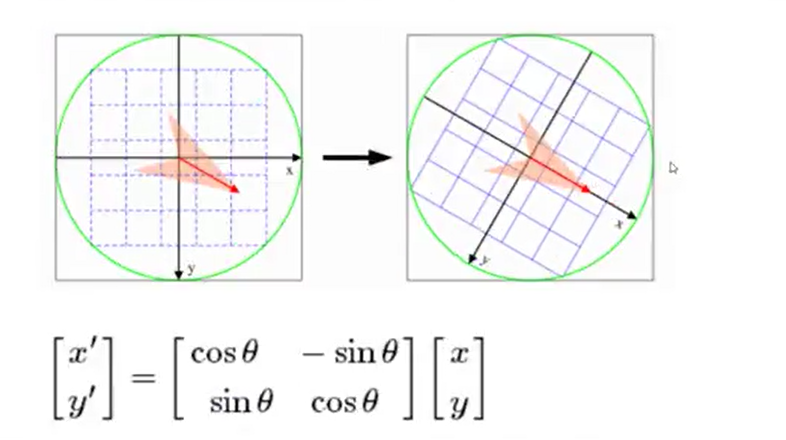
第七步：使用sobel算子，每个特征点得到三个信息，获得位置， 计算梯度的大小，以及梯度的方向



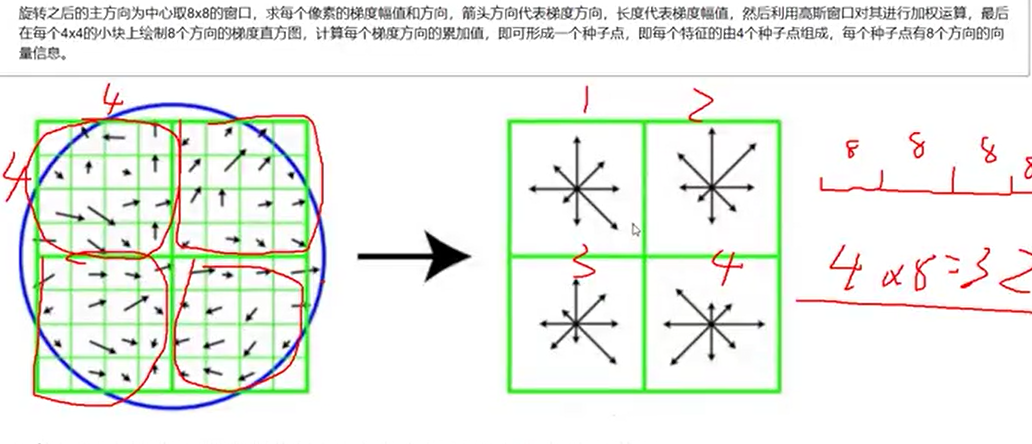
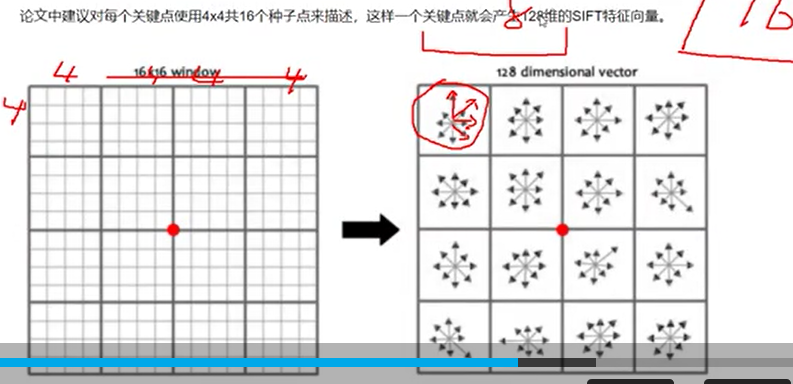
第八步：统计相邻部分的梯度的方向，画出直方图，把直方图中出现次数最多的作为主方向，如果次方向的次数大于主方向的0.8，那么次方向也是辅助方向



 第九步：将梯度的方向进行按照原来的方向进行旋转，以保证梯度旋转的不变性



第十步：对特征点进行领域的位置统计，来生成sift特征向量， 对于左边的那个图，从4\*4个领域中统计八个方向，因此有4\*8个sift，右边有16个即16\*8=128个特征向量

 代码：

第一步：读入图片

第二步：进行灰度化

第三步：使用cv2.xfeatures2d.SIFT\_create() 实例化sift函数

第四步：使用sift.detect(gray, None) 生成关键点

第五步：使用cv2.drawKeypoints 进行画图操作

第六步：使用sift.compute(kp) 求得关键点对应的128个特征向量

[复制代码](javascript:void(0);)

import numpy as np

import cv2

img = cv2.imread('test\_1.jpg')

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

sift = cv2.xfeatures2d.SIFT\_create()

# 找出关键点

kp = sift.detect(gray, None)

# 对关键点进行绘图

ret = cv2.drawKeypoints(gray, kp, img)

cv2.imshow('ret', ret)

cv2.waitKey(0)

cv2.destroyAllWindows()

# 使用关键点找出sift特征向量

kp, des = sift.compute(gray, kp)

print(np.shape(kp))

print(np.shape(des))

print(des[0])

# 二、利用sift特征进行特征匹配

# [机器学习进阶-案例实战-图像全景拼接-书籍SIFT特征点连接 1.cv2.drawMatches(对两个图像的关键点进行连线操作)](https://www.cnblogs.com/my-love-is-python/p/10422152.html)

1.cv2.drawMatches(imageA, kpsA, imageB, kpsB, matches[:10], None, flags=2)  # 对两个图像关键点进行连线操作

参数说明：imageA和imageB表示图片，kpsA和kpsB表示关键点， matches表示进过cv2.BFMatcher获得的匹配的索引值，也有距离， flags表示有几个图像

书籍的SIFT特征点连接：

   第一步:使用sift.detectAndComputer找出关键点和sift特征向量

   第二步：构建BFMatcher()蛮力匹配器，bf.match匹配sift特征向量，使用的是欧式距离

   第三步：根据匹配结果matches.distance对matches按照距离进行排序

   第四步：进行画图操作，使用cv2.drawMatches进行画图操作

[复制代码](javascript:void(0);)

import cv2

import numpy as np

#读入图片

imgA = cv2.imread('box.png', 0)

imgB = cv2.imread('box\_in\_scene.png', 0)

def cv\_show(img, name):

cv2.imshow(name, img)

cv2.waitKey(0)

cv2.destroyAllWindows()

# 第一步：构造sift，求解出特征点和sift特征向量

sift = cv2.xfeatures2d.SIFT\_create()

kpsA, dpA = sift.detectAndCompute(imgA, None)

kpsB, dpB = sift.detectAndCompute(imgB, None)

# 第二步：构造BFMatcher()蛮力匹配，匹配sift特征向量距离最近对应组分

bf = cv2.BFMatcher()

# 获得匹配的结果

matches = bf.match(dpA, dpB)

#第三步：对匹配的结果按照距离进行排序操作

matches = sorted(matches, key=lambda x: x.distance)

# 第四步：使用cv2.drawMacthes进行画图操作

ret = cv2.drawMatches(imgA, kpsA, imgB, kpsB, matches[:10], None, flags=2)

cv2.imshow('ret', ret)

cv2.waitKey(0)

cv2.destroyAllWindows()

[复制代码](javascript:void(0);)

