**Python-opencv3 SIFT算法做特征匹配**

最近接触一个项目：根据设计师定出的psd格式文件（photoshop），生成不同尺寸的海报。这里面牵扯到了尺度不变而对特征做变换的问题。这里简单介绍一下SIFT的概念，并知晓如何找到SIFT中的Keypoints和Descriptors，最后展示一个Demo。

**一点推荐**

作为CSDN的忠实用户，最近发现CSDN学院上了一些对新手比较友好的课程。以我的切身体会来看，对于想要了解机器学习算法或者python编程语言的同学，非常有帮助。还记得我最开始学习python的时候，看的是一本写给小孩子的书《**趣学Python——教孩子学编程**》。

虽然这本书不错，但是确实有些过于简单了，而CSDN提供的课程有两门对现在的我来讲还是有相当大的帮助，老师讲课水平高，配合丰富的例子，容易让人掌握知识点，下面推荐两门课程：

**人工智能在网络领域的应用与实践**:  
<https://edu.csdn.net/course/play/10319?utm_source=sooner>

ps: 如果想要系统学习python的朋友，下面这门课是涵盖了python基础语法、web开发、数据挖掘以及机器学习，是CSDN强力推荐的课程，有需要的朋友可以看看哈：

**Python全栈工程师**:  
<https://edu.csdn.net/topic/python115?utm_source=sooner>

**简介**

**SIFT**（[Scale](https://so.csdn.net/so/search?q=Scale&spm=1001.2101.3001.7020)-invariant feature transform），也叫**尺度不变特征变换算法**，是David Lowe于1999年提出的**局部特征描述子**（Descriptor），并于2004年进行了更深入的发展和完善。Sift特征匹配算法可以处理两幅图像之间发生平移、旋转、仿射变换情况下的匹配问题，具有很强的匹配能力。在Mikolajczyk对包括Sift算子在内的十种局部描述子所做的不变性对比实验中，Sift及其扩展算法已被证实在同类描述子中具有最强的健壮性。

其应用范围包含物体辨识、机器人地图感知与导航、影像缝合、3D模型建立、手势辨识、影像追踪和动作比对。局部影像特征的描述与侦测可以帮助辨识物体，SIFT 特征是基于物体上的一些局部外观的兴趣点而与影像的大小和旋转无关。对于光线、噪声、些微视角改变的容忍度也相当高。基于这些特性，它们是高度显著而且相对容易撷取，在母数庞大的特征数据库中，很容易辨识物体而且鲜有误认。

**SIFT算法特点与步骤**

* **Sift特征是图像的局部特征** 对平移、旋转、尺度缩放、亮度变化、遮挡和噪声等具有良好的不变性，对视觉变化、仿射变换也保持一定程度的稳定性。
* **独特性好** 信息量丰富，适用于在海量特征数据库中进行快速、准确的匹配。
* **多量性** 即使少数的几个物体也可以产生大量Sift特征向量。
* **速度相对较快** 经优化的Sift匹配算法甚至可以达到实时的要求。
* **可扩展性强** 可以很方便的与其他形式的特征向量进行联合。

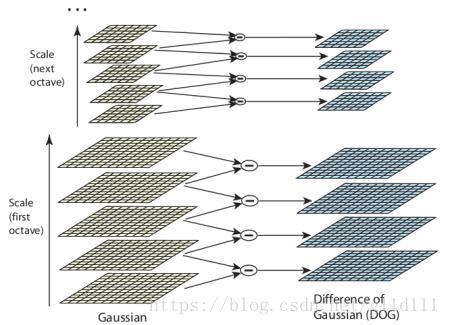
SIFT算法的实质是：**“不同的尺度空间上查找关键点(特征点)，并计算出关键点的方向”** ，SIFT所查找到的关键点是一些十分突出，不会因光照，仿射变换和噪音等因素而变化的点，如角点、边缘点、暗区的亮点及亮区的暗点等。

Lowe将SIFT算法分解为如下四步：

* **① 尺度空间极值检测**：搜索所有尺度上的图像位置。通过*高斯微分函数*来识别潜在的对于尺度和旋转不变的兴趣点。

为了在不同的尺度空间检测关键点。这里，尺度空间的获取需要使用尺度空间滤波（scale-space filtering）来实现，Lindeberg等人已证明高斯卷积核是实现尺度变换的唯一变换核，并且是唯一的线性核。尺度规范化的**LoG**(Laplacion of Gaussian)算子具有真正的尺度不变性，但是由于LoG算法复杂度较高。因此，Lowe使用更为高效的**高斯差分算子**（Difference of Gaussians）近似LoG算子来进行极值检测，如下：  
D ( x , y , σ ) = ( G ( x , y , k σ ) − G ( x , y , σ ) ) ∗ I ( x , y ) = L ( x , y , k σ ) − L ( x , y , σ ) D(x,y,\sigma) = (G(x,y,k\sigma) - G(x,y,\sigma) ) \* I(x,y) = L(x,y,k\sigma) - L(x,y,\sigma)*D*(*x*,*y*,*σ*)=(*G*(*x*,*y*,*kσ*)−*G*(*x*,*y*,*σ*))∗*I*(*x*,*y*)=*L*(*x*,*y*,*kσ*)−*L*(*x*,*y*,*σ*)

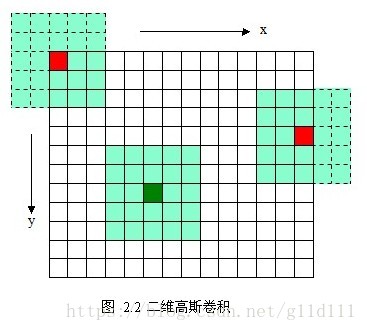
由上式可以看出，**高斯差分算子**（Difference of Gaussians）是使用两个不同的σ \sigma*σ*，k σ k\sigma*kσ*来做高斯模糊差异而得到的。这里，∗ \*∗表示卷积操作，G ( x , y , σ ) G(x,y,\sigma)*G*(*x*,*y*,*σ*)为一个变化尺度的**高斯**（Gaussian ）函数，I ( x , y ) I(x,y)*I*(*x*,*y*)表示原图像。

在实际计算时，使用**高斯金字塔**（Gaussian Pyramid）每组中相邻上下两层图像相减，得到高斯差分图像，如下图所示（关于高斯金字塔中的构建在**内容延伸**中会介绍）：  


G ( x , y , σ ) = 1 2 π σ 2 e − ( x − m / 2 ) 2 + ( y − n / 2 ) 2 2 σ 2 G(x,y,\sigma) = \frac{1}{2\pi\sigma^2}e^-\frac{(x - m/2)^2 + (y - n/2)^2}{2\sigma^2}*G*(*x*,*y*,*σ*)=2*πσ*21​*e*−2*σ*2(*x*−*m*/2)2+(*y*−*n*/2)2​  
m , n m,n*m*,*n*表示高斯模板的维度(由( 6 σ + 1 ) ( 6 σ + 1 ) (6\sigma +1)(6\sigma + 1)(6*σ*+1)(6*σ*+1)确定。x , y x, y*x*,*y*代表图像的像素位置。σ \sigma*σ*是尺度空间因子，值越小表示图像被平滑的越少，相应的尺度也就越小。**大尺度对应于图像的结构，小尺度对应于图像的细节纹理特征。**

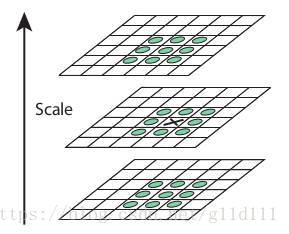
下面3张图转自博客：<https://blog.csdn.net/zddblog/article/details/7521424>





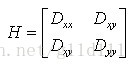
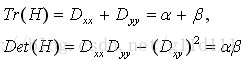


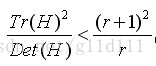
当我们得到DoG（Difference of Gaussian）之后，图像在尺度空间中搜寻**局部极值**（local extrema）。以下图为例，在图像中的某个像素点不但与其附近的8个像素点比较，而且与其前一层（previous scale）的9个像素点和下一层（next scale）的9个像素点进行比较（需为同一Octave）。如果该像素点是局部极值点，那么我们就认为它是一个潜在的**KeyPoint**（关键点）——最能代表这个scale的点

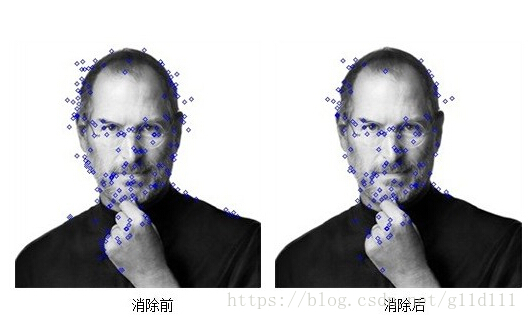


论文中给出了一些经验的值：octave（组）为4，scale layer（层）为5，σ = 1.6 \sigma=1.6*σ*=1.6，k = 2 k=\sqrt 2*k*=2​。

* **② 关键点定位**：在每个候选的位置上，通过一个拟合精细的模型来确定位置和尺度。关键点的选择依据于它们的稳定程度。  
  由第①步检测得到的极值点是离散空间的极值点。下面，通过拟合三维二次函数来精确确定关键点的位置和尺度，同时去除**低对比度的关键点**和**不稳定的边缘响应点**(因为DoG算子会产生较强的边缘响应)，以增强匹配稳定性、提高抗噪声能力。  
  他们使用泰勒级数展开式空间来获得更精确的极值位置，如果这个极值点的强度小于阈值，它就会被拒绝。这个阈值在Opencv中为**contrastThreshold**。

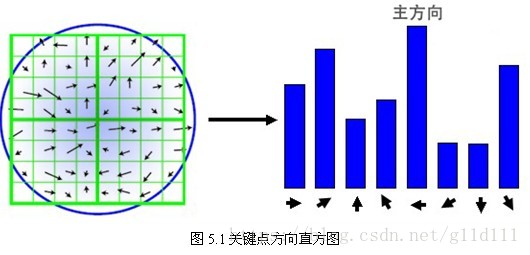
因为DoG算子会产生较强的边缘响应，所以要去除这些不稳定的边缘响应点。与Harris角点检测的思路相似，获取特征点处的Hessian矩阵，**主曲率**（principal curvature）通过一个2x2 的Hessian矩阵H H*H*求出：  
  
H H*H*的特征值α α*α*和β β*β*分别代表x和y方向的梯度，  
  
这里，T r ( H ) Tr(H)*Tr*(*H*)为主对角线元素之和，D e t ( H ) Det(H)*Det*(*H*)表示矩阵H的行列式。假设是α α*α*较大的特征值，而β β*β*为较小的特征值，令α = r β α=rβ*α*=*rβ*，则  
这里写图片描述

D D*D*的主曲率和H H*H*的特征值成正比，跟上面一样，让α α*α*为最大特征值，β β*β*为最小特征值，则公式( r + 1 ) 2 r \frac{(r+1)^2}{r}*r*(*r*+1)2​的值在两个特征值相等时最小，随着r r*r*的增大而增大。r r*r*值越大，说明两个特征值的比值越大，即在某一个方向的梯度值越大，而在另一个方向的梯度值越小，边缘恰恰就是一个方向梯度大，一个方向梯度小的情况。所以为了剔除边缘响应点，需要让该比值小于一定的阈值（论文给的r r*r*值为10）：  
  
将满足条件的关键点保留，否则则扔掉。因此，低对比度和边缘的关键点在本步骤中被去除，只保留最感兴趣的那些点。

下图左为边缘消除前的关键点分布图，右是消除后的，这里可以看出一些边缘角点被去除掉了。  


* **③ 方向确定**：基于图像局部的梯度方向，分配给每个关键点位置一个或多个方向。所有后面的对图像数据的操作都相对于关键点的方向、尺度和位置进行变换，从而提供对于这些变换的不变性。

为了使**描述符**（Descriptors）具有旋转不变性，需要利用图像的局部特征为给每一个关键点分配一个基准方向。使用图像梯度的方法求取局部结构的稳定方向。对于在DoG中检测出的关键点，采集其所在高斯金字塔图像3 σ 3σ3*σ*邻域窗口内像素的梯度和方向分布特征。梯度的模值和方向如下：  
这里写图片描述  
L为关键点所在的尺度空间值，按Lowe的建议，梯度的模值m(x,y)按的σ = 1.5 σ o c t \sigma = 1.5\sigma\_{oct}*σ*=1.5*σoct*​高斯分布，按尺度采样的3 σ 3σ3*σ*原则，邻域窗口半径为3 × 1.5 σ o c t 3\times1.5\sigma\_{oct}3×1.5*σoct*​。  
在完成关键点的梯度计算后，使用直方图统计邻域内像素的梯度和方向。梯度直方图将0~360度的方向范围分为36个柱(bins)，其中每柱10度。如下图所示，直方图的峰值方向代表了关键点的主方向，(为简化，图中只画了八个方向的直方图)。

  
方向直方图的峰值则代表了该特征点处邻域梯度的方向，**以直方图中最大值作为该关键（KeyPoint）的主方向**。为了增强匹配的鲁棒性，只保留峰值大于主方向峰值80％的方向作为该关键点的辅方向。因此，对于同一梯度值的多个峰值的关键点位置，在相同位置和尺度将会有多个关键点被创建但方向不同。仅有15％的关键点被赋予多个方向，但可以明显的提高关键点匹配的稳定性。

实际编程实现中，就是把该关键点复制成多份关键点，并将方向值分别赋给这些复制后的关键点，并且，离散的梯度方向直方图要进行插值拟合处理，来求得更精确的方向角度值，检测结果如图所示。  


至此，将检测出的含有位置、尺度和方向的关键点即是该图像的SIFT特征点。

* **④ 关键点描述**：在每个关键点周围的邻域内，在选定的尺度上测量图像局部的梯度。这些梯度被变换成一种表示，这种表示允许比较大的局部形状的变形和光照变化。

**Demo**

* 基于python3.x + opencv3.x

import cv2

import numpy as np

from psd\_tools import PSDImage

# 1) psd to png

psd1 = PSDImage.load('200x800.ai.psd')

psd1.as\_PIL().save('psd\_image\_to\_detect1.png')

psd2 = PSDImage.load('800x200.ai.psd')

psd2.as\_PIL().save('psd\_image\_to\_detect2.png')

# 2) 以灰度图的形式读入图片

psd\_img\_1 = cv2.imread('psd\_image\_to\_detect1.png', cv2.IMREAD\_GRAYSCALE)

psd\_img\_2 = cv2.imread('psd\_image\_to\_detect2.png', cv2.IMREAD\_GRAYSCALE)

# 3) SIFT特征计算

sift = cv2.xfeatures2d.SIFT\_create()

psd\_kp1, psd\_des1 = sift.detectAndCompute(psd\_img\_1, None)

psd\_kp2, psd\_des2 = sift.detectAndCompute(psd\_img\_2, None)

# 4) Flann特征匹配

FLANN\_INDEX\_KDTREE = 1

index\_params = dict(algorithm=FLANN\_INDEX\_KDTREE, trees=5)

search\_params = dict(checks=50)

flann = cv2.FlannBasedMatcher(index\_params, search\_params)

matches = flann.knnMatch(psd\_des1, psd\_des2, k=2)

goodMatch = []

for m, n in matches:

# goodMatch是经过筛选的优质配对，如果2个配对中第一匹配的距离小于第二匹配的距离的1/2，基本可以说明这个第一配对是两幅图像中独特的，不重复的特征点,可以保留。

if m.distance < 0.50\*n.distance:

goodMatch.append(m)

# 增加一个维度

goodMatch = np.expand\_dims(goodMatch, 1)

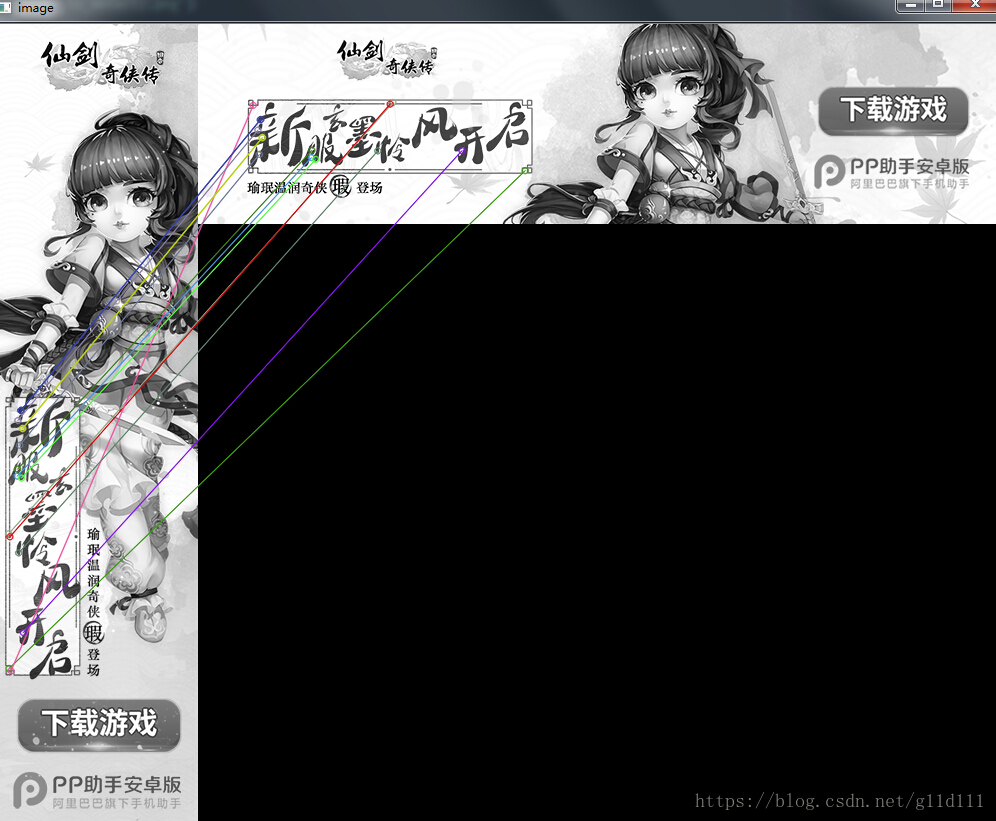
print(goodMatch[:20])

img\_out = cv2.drawMatchesKnn(psd\_img\_1, psd\_kp1, psd\_img\_2, psd\_kp2, goodMatch[:15], None, flags=2)

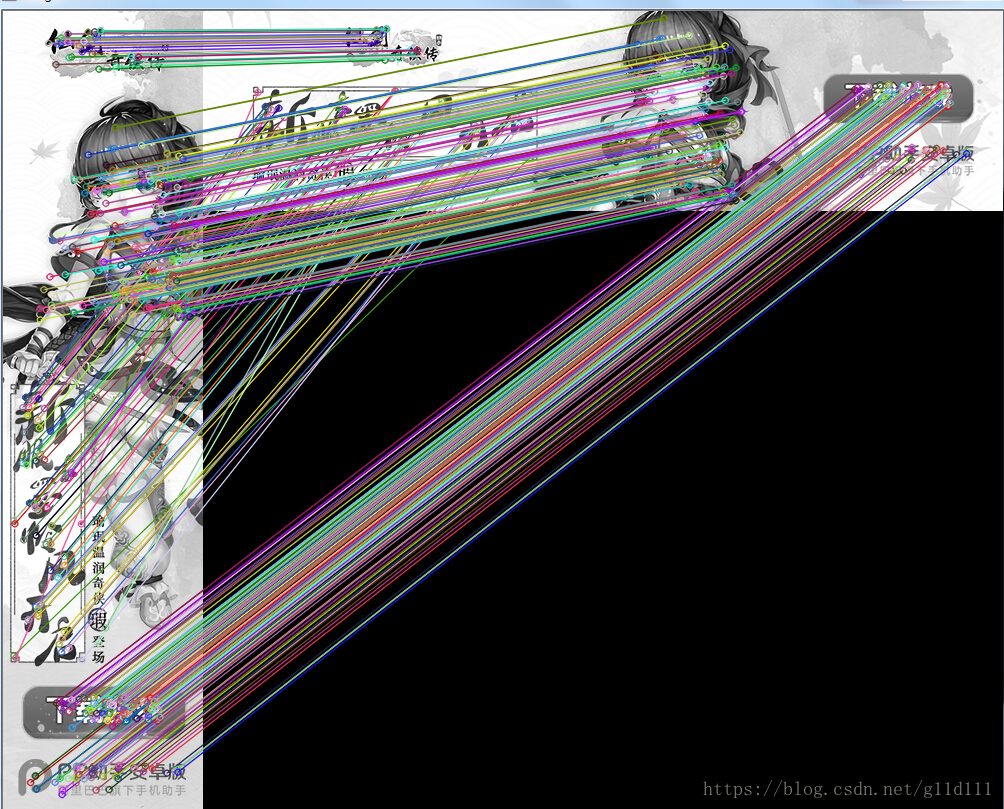
cv2.imshow('image', img\_out)#展示图片

cv2.waitKey(0)#等待按键按下

cv2.destroyAllWindows()#清除所有窗口



如果把img\_out中的参数goodMatch[:15]改为goodMatch，即展示所有匹配的特征，图为：

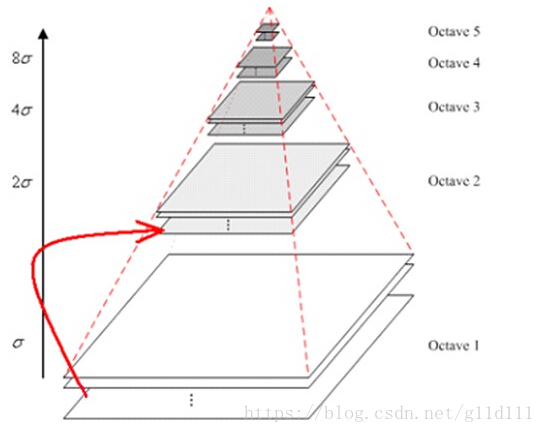


**内容延伸**

**1 高斯金字塔的构建**

在**SIFT算法特点与步骤**的第一部分中，尺度空间在实现时使用高斯金字塔（Gaussian Pyramid）表示，这里介绍一下高斯金字塔的构造过程。高斯金字塔的构建分为两部分：

* 对图像做不同尺度（不同σ \sigma*σ*）的高斯模糊；
* 对图像做降采样(隔点采样)。



图像的金字塔模型是指：“将原始图像不断**降采样**（Downsampling），得到一系列大小不一的图像，由大到小，从下到上构成的塔状模型。” ，原图像为金子塔的最底层（Octave 1），每次降采样所得到的新图像为金字塔的一层。金字塔的层数根据图像的原始大小和塔顶图像的大小共同决定，其计算公式如下：

这里写图片描述

其中M , N M,N*M*,*N*为原图像的大小，t t*t*为塔顶图像的最小维数的对数值。如，对于大小为512*512的图像，金字塔上各层图像的大小如下表所示，当塔顶图像为4*4时，n=7，当塔顶图像为2\*2时，n=8。  


octave–组，Layer–层，scale–尺度。为了让尺度体现其连续性，高斯金字塔在简单降采样的基础上加上了高斯滤波。如上图所示，将图像金字塔每层的一张图像使用**不同尺度**（scale）做高斯模糊，使得金字塔的每层含有多张高斯模糊图像，将金字塔每层的多张图像合称为一组(Octave)，金字塔每层只有一组图像，组数和金字塔层数相等。

注意：为了在每组中检测S个尺度的极值点，则DoG金字塔每组需S+2层图像，而DoG金字塔由高斯金字塔同组的相邻两层相减得到，则高斯金字塔每组需S+3层图像，实际计算时S在3到5之间。lowe推荐S取3。

1. [Introduction to SIFT (Scale-Invariant Feature Transform)](https://docs.opencv.org/3.1.0/da/df5/tutorial_py_sift_intro.html)
2. [Distinctive Image Features from Scale-Invariant Keypoints——David G. Lowe](https://link.springer.com/article/10.1023/B:VISI.0000029664.99615.94)
3. [SIFT算法详解——zddhub](https://blog.csdn.net/zddblog/article/details/7521424)