Sift进一步学习

2016年10月19日　　18:11

留存问题：
1.DOG局部极值检测，关键点精确定位。关键点与D(X)极值点的关系。
2.关键点特征向量的生成，以及主方向的确定。关键点的主方向确定与种子点之间的关系。

DOG高斯差分算子的直观解释：
在计算机视觉中，高斯差（英语：Difference of Gaussians，简称“DOG”）是一种将一个原始灰度图像的模糊图像从另一幅灰度图像进行增强的算法，通过DOG以降低模糊图像的模糊度。这个模糊图像是通过将原始灰度图像经过带有不同标准差的高斯核进行卷积得到的。用高斯核进行高斯模糊只能压制高频信息。从一幅图像中减去另一幅可以保持在两幅图像中所保持的频带中含有的空间信息。这样的话，DOG就相当于一个能够去除除了那些在原始图像中被保留下来的频率之外的所有其他频率信息的带通滤波器。

尺度变化连续性：
假设s=3，也就是每个塔里有3层，则$k=2^{1/s}=2^{1/3}$，那么按照上图可得Gauss Space和DoG space分别有3个（s个）和2个（s-1个）分量，在DoG space中，1st-octave两项分别是σ,kσ; 2nd-octave两项分别是2σ,2kσ;由于无法比较极值，我们必须在高斯空间继续添加高斯模糊项，使得形成$\sigma, k\sigma, k^2\sigma, k^3\sigma, k^4\sigma$这样就可以选择DoG space中的中间三项$k\sigma, k^2\sigma, k^3\sigma$（只有左右都有才能有极值），那么下一octave中（由上一层降采样获得）所得三项即为$2k\sigma, 2k^2\sigma, 2k^3\sigma$，其首项$2k\sigma=2^{4/3}$。刚好与上一octave末项$k^3\sigma=2^{3/3}$尺度变化连续起来，所以每次要在Gaussian space添加3项，每组（塔）共S+3层图像，相应的DoG金字塔有S+2层图像。

来自 <http://blog.csdn.net/abcjennifer/article/details/7639681>

Sift特征检测的4个主要步骤：

1. **尺度空间的极值检测** 搜索所有尺度空间上的图像，通过高斯微分函数来识别潜在的对尺度和选择不变的兴趣点。
2. **特征点定位** 在每个候选的位置上，通过一个拟合精细模型来确定位置尺度，关键点的选取依据他们的稳定程度。
3. **特征方向赋值** 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，后续的所有操作都是对于关键点的方向、尺度和位置进行变换，从而提供这些特征的不变性。
4. **特征点描述** 在每个特征点周围的邻域内，在选定的尺度上测量图像的局部梯度，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换。

来自 <http://www.cnblogs.com/wangguchangqing/p/4853263.html>

拉普拉斯算子：

定义

编辑

拉普拉斯算子是n维欧几里德空间中的一个二阶微分算子，定义为梯度（▽f）的散度（▽·f）。因此如果f是二阶可微的实函数，则f的拉普拉斯算子定义为：

$$\Delta f = \nabla^2 f = \nabla \cdot \nabla f,$$

f的拉普拉斯算子也是笛卡儿坐标系x_i中的所有*非混合*二阶偏导数：

$$\Delta f = \sum_{i=1}^{n} \frac{\partial^2 f}{\partial x_i^2}.$$

作为一个二阶微分算子，拉普拉斯算子把C函数映射到C函数，对于$k \geq 2$。表达式(1)（或(2)）定义了一个算子$\Delta: C(\mathbf{R}) \to C(\mathbf{R})$，或更一般地，定义了一个算子$\Delta: C(\Omega) \to C(\Omega)$，对于任何开集$\Omega$。

函数的拉普拉斯算子也是该函数的黑塞矩阵的迹 $(x+a)^n = \sum_{k=0}^{n} \binom{n}{k} x^k a^{n-k}$：

$$\Delta f = \mathrm{tr}(H(f)).$$

另外，满足▽·▽f=0 的函数f，称为调和函数.

Python+opencv代码：

```
import cv2
import numpy as np

#import pdb
#pdb.set_trace()#turn on the pdb prompt


#read image

img = cv2.imread(
'D:\privacy\picture\little girl.jpg',cv2.IMREAD_COLOR)
gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

cv2.imshow(
'origin',img);


#SIFT
detector = cv2.SIFT()
```

```
keypoints = detector.detect(gray, None)
img = cv2.drawKeypoints(gray,keypoints)

#img = cv2.drawKeypoints(gray,keypoints,flags = cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
cv2.imshow('test',img);

cv2.waitKey(0)
cv2.destroyAllWindows()
```

来自 <http://blog.csdn.net/abcjennifer/article/details/7639681>