SIFT算法

SIFT核心知识点

尺度空间

> 尺度空间理论的基本思想是：在图像信息处理模型中引入一个被视为尺度的参数，通过连续变化尺度参数获得多尺度下的尺度空间表示序列，对这些序列进行尺度空间主轮廓的提取。

> 尺度空间满足视觉不变性。该不变性的视觉解释如下：当我们用眼睛观察物体时，一方面当物体所处背景的光照条件变化时，视网膜感知图像的亮度水平和对比度是不同的，因此要求尺度空间算子对图像的分析不受图像的灰度水平和对比度变化的影响，即满足灰度不变性和对比度不变性。另一方面，相对于某一固定坐标系，当观察者和物体之间的相对位置变化时，视网膜所感知的图像的位置、大小、角度和形状是不同的，因此要求尺度空间算子对图像的分析和图像的位置、大小、角度以及仿射变换无关，即满足平移不变性、尺度不变性、欧几里德不变性以及仿射不变性。（来源于：https://blog.csdn.net/dcrmg/article/details/52561656）

在SIFT中，所谓尺度空间实际上就是由高斯金字塔以及高斯差分金字塔中的不同组，不同层之间的尺寸不同，模糊程度不同的图像构成，也就是(O, S, σ)构成，相当于上述人眼视物中，同一个物体，由于相对距离产生的尺寸变小，轮廓变得模糊的过程。

高斯核与高斯卷积

高斯核（关于高斯核，Lindeberg证明其为实现尺度变换的唯一变换核，其余核会对图像造成模糊之外的影响，当做结论性理论吧。）

$$G(r)=\frac{1}{\sqrt{2\pi\sigma^2}^N}e^{-r^2/(2\sigma^2)}$$

高斯卷积

$$L(x,y,\sigma)=G(x,y,\sigma)*I(x,y)$$

$$G(x,y,\sigma)=\frac{1}{2\pi\sigma^2}e^{-\frac{(x-m/2)^2+(y-n/2)^2}{2\sigma^2}}$$

高斯卷积实际上是做了一个对图像做了加权平均模糊滤波，在SIFT中 σ 是被称为尺度因子，我的理解是，它控制了图像在尺度变换过程造成了细节模糊。不同尺度下所共同具有的特征，才是稳定的特征。

高斯金字塔

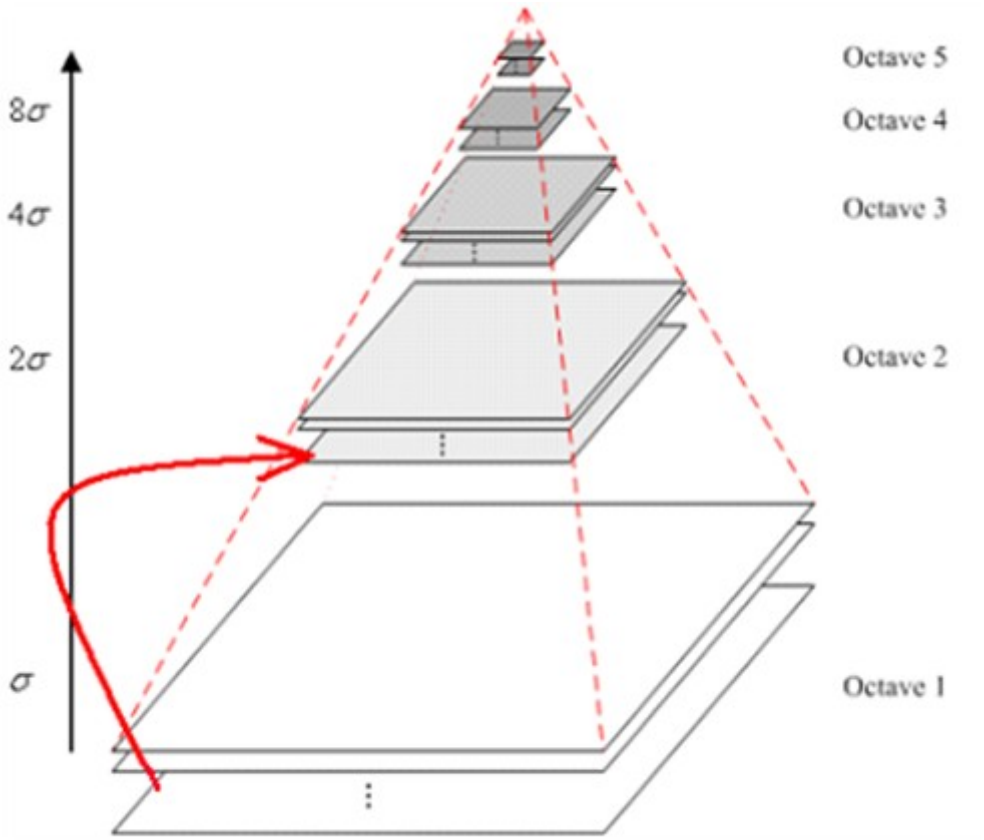


图 3.1 高斯金字塔

上图所示，是高斯金字塔的模型，至下而上，是0组，1组……尺寸变化为1/2倍的迭代，每一组之间由不同的σ 值实现不同的模糊程度。

DOG金字塔

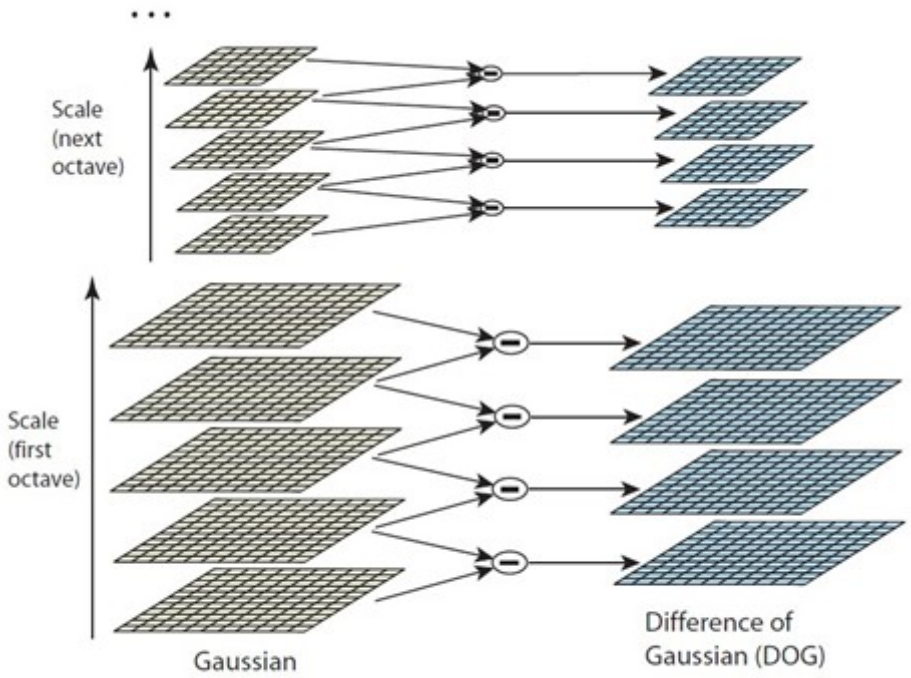


图 3.3 高斯差分金字塔的生成

高斯差分涉及到尺度归一化的高斯拉普拉斯函数$\sigma^2\nabla^2 G$,高斯差分实际上是做了一个离散数字序列与拉普拉斯的相似替换。

上图清晰的表达了高斯金字塔与高斯差分金字塔的关系，高斯金字塔中同组图像，相邻两层相减，形成DOG（高斯差分金子塔），自此已经提取了基本的梯度信息。

空间极值点检测

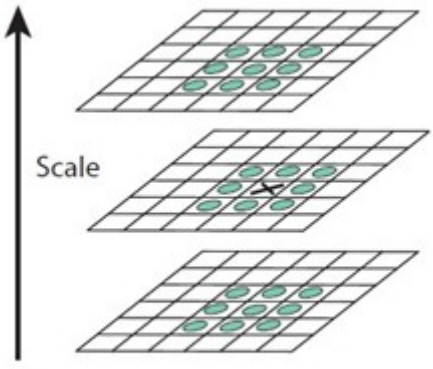


图 3.4 DOG 空间极值检测

为了寻找DoG函数的极值点，每一个像素点要和它所有的相邻点比较，看其是否比它的图像域和尺度域的相邻点大或者小。如图3.4所示，中间的检测点和它同尺度的8个相邻点和上下相邻尺度对应的9×2个点共26个点比较，以确保在尺度空间和二维图像空间都检测到极值点。

手动实现高斯金字塔

定义了3层高斯金字塔，定义了-1层，也就是最低层采用了上采样，将图像扩大了一倍，并进行高斯平滑

效果图

实现代码

```
# -*- coding: utf-8 -*-

"""
@author: liuxin
@contact: xinliu1996@163.com
@Created on: 2019/10/20 18:08
"""

class GaussPyramindLayer(object):
    """
    create gauss pyramid
    """
    def __init__(self,image_src=None,sigma=1):
        self.image_src=image_src
        self.sigma = sigma
```

```
    def create_layer(self):
        filter = self.gauss_kernel(5,self.sigma)
        img = self.conv_2d(filter,self.image_src)
        return img

    # 二维高斯滤波核函数
    def gauss_kernel(self,kernel_size, sigma):
        kernel = np.zeros((kernel_size, kernel_size))
        s = sigma ** 2
        center = kernel_size // 2
        sum_val = 0
        for i in range(kernel_size):
            for j in range(kernel_size):
                x, y = i - center, j - center
                kernel[i, j] = np.exp(-(x ** 2 + y ** 2) / (2 * s))
                sum_val += kernel[i, j]
        kernel = kernel / sum_val
        return kernel

    # 卷积函数
    def conv_2d(self,kernel, img, mode='fill'):
        if mode == 'fill':
            h = kernel.shape[0] // 2
            w = kernel.shape[1] // 2
            img = np.pad(img, (h, w), 'reflect')

        res_h = img.shape[0] - kernel.shape[0] + 1
        res_w = img.shape[1] - kernel.shape[1] + 1

        res = np.zeros((res_h, res_w))

        dh = kernel.shape[0]
        dw = kernel.shape[1]
        for i in range(res_h):
            for j in range(res_w):
                res[i, j] = np.sum(img[i:i + dh, j:j + dw] * kernel)
        return res



src = cv.imread('ground.jpeg',0)
src = cv.resize(src,(512,512))
cv.imshow('src', src)
#定义组的数量以及组内层数
octaves_layers = 3
internal_layers =3
sigma0 =1.6
# 先对图像上采样，也就是-1层，利用内插法上采样
image = cv.pyrUp(src,dstsize=(src.shape[0]*2,src.shape[0]*2))
for i in range(octaves_layers):
    #下采样
    image_shape = image.shape[0]
    image = cv.pyrDown(image)
    for j in range(internal_layers):
        sigma_i_j = 2**(i+j/internal_layers)
        print(sigma_i_j)
        layer = GaussPyramindLayer(image_src=image,sigma=sigma_i_j)
        image=layer.create_layer()
```

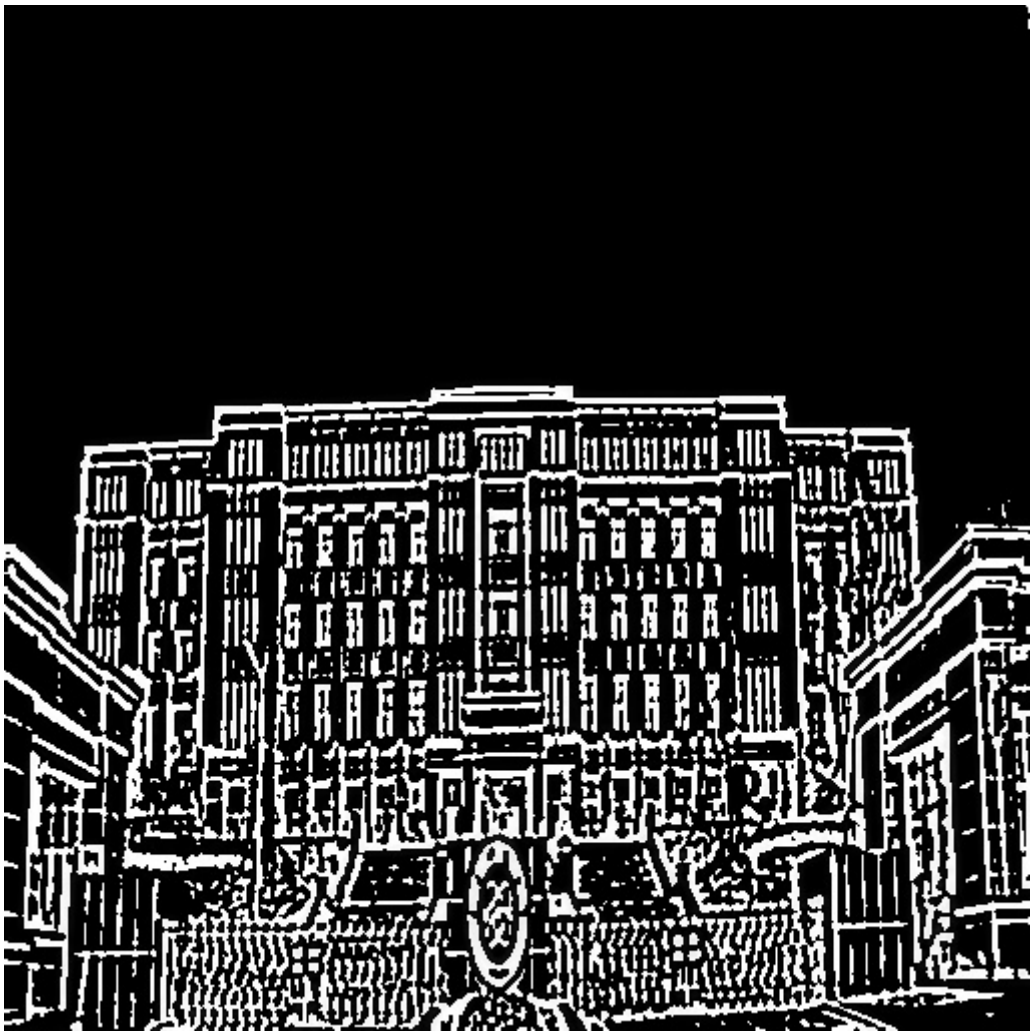
```
        name='layer_{0}_{1}.jpg'.format(i,j)
        cv.imwrite(name,image.astype(dtype=np.uint8))


cv.waitKey(0)
cv.destroyAllWindows()
```

手动实现高斯差分（DOG）金字塔

这里受限于篇幅，我每一组只做了两层相减，也就是只产生一层DOGlayer

效果图



核心实现代码：

```
for i in range(octaves_layers):
    #下采样
    image_shape = image.shape[0]
    image = cv.pyrDown(image)
    image_list = []
    for j in range(internal_layers):
        sigma_i_j = 2**(i+j/internal_layers)
        print(sigma_i_j)
        layer = GaussPyramindLayer(image_src=image,sigma=sigma_i_j)
        image=layer.create_layer()
        image_list.append(image.copy())
        name='layer_{0}_{1}.jpg'.format(i,j)
        cv.imwrite(name,image.astype(dtype=np.uint8))
    DOG_image =  image_list[0]-image_list[1]

    name = 'DOG_layer{0}.jpg'.format(i)
    cv.imwrite(name, DOG_image.astype(dtype=np.uint8))
```

利用Opencv实现SIFT

实在是很多东西还不理解，手动实现整个SIFT过程还有些困难。

效果图：

代码：

```
# -*- coding: utf-8 -*-

"""
@author: liuxin
@contact: xinliu1996@163.com
```

```
@Created on: 2019/10/20 18:08
"""
import numpy as np
import cv2
from matplotlib import pyplot as plt

imgname1 = 'ground.jpeg'

sift = cv2.xfeatures2d.SIFT_create(sigma=1.6)

img1 = cv2.imread(imgname1)
gray1 = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY) #灰度处理图像
kp1, des1 = sift.detectAndCompute(img1,None)   #des是描述子

img3 = cv2.drawKeypoints(img1,kp1,img1,color=(255,0,255)) #画出特征点，并显示为红色圆圈

cv2.imwrite('sift_point.jpg',img3)
cv2.waitKey(0)
cv2.destroyAllWindows()
```