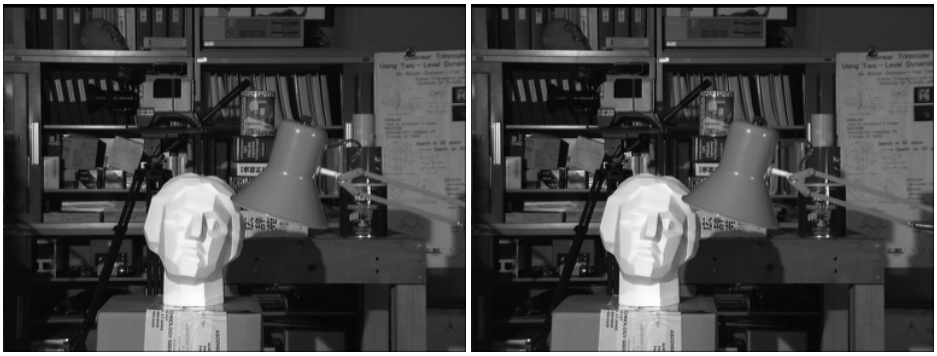
双目三维重建实验报告

蒋颜丞，自动化(电气)1903，3190102563

1 实验内容和要求

针对附件图像对（相机平行配置），编写代码实现双目三维重建过程，计算视差图。要求提供代码和结果视差图。



左图：左相机照片；右图：右相机照片

2 实验原理

2.1 窗口搜索法

对于左图的每一待匹配点，取其window_size*window_size大小的邻域作为窗口(region A)，在右图的同一水平极线上，在该点坐标的左侧一定范围内使用滑动窗口(region B)计算各点与待匹配点的NCC值，取最大NCC值对应的滑动步长作为视差值。

2.2 NCC匹配

NCC（Normalization Cross Correlation，归一化相关性），用于表示归一化待匹配目标之间的相关程度。对于原始的图像内任意一个像素点(px, py)构建一个$n \times n$的邻域作为匹配窗口。然后对于目标相素位置$(px+d, py)$同样构建一个$n \times n$大小的匹配窗口，对两个窗口进行相似度度量，这里的d有一个取值范围。NCC 计算公式如下：

$$NCC\left(I_1, I_2\right)=\frac{\sum_x\left(I_1(x)-\mu_1\right)\left(I_2(x)-\mu_2\right)}{\sqrt{\sum_x\left(I_1(x)-\mu_1\right)^2 \sum_x\left(I_2(x)-\mu_2\right)}}$$

其中，$I_1(x)$为原始图像的像素值，μ_1为原始窗口内像素的均值，$I_2(x)$为原始图像在目标图像上对应点位置在x方向上偏移d后的像素值，μ_2为目标图像匹配窗口像素均值。当$NCC=1$时，两个匹配窗口的图片信息具有很高的相关性，当$NCC=-1$时，则说明两个匹配窗口的图片信息完全不相关。

3 源代码

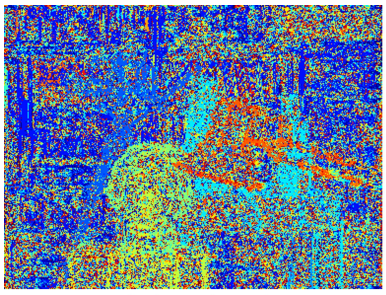
```python
import cv2
import numpy as np
from scipy.ndimage import filters

def plane_sweep_gauss(img_l, img_r, start, steps, size):
    '''
```

```
    采用平面滑动的方式、使用带高斯加权的归一化相关性指标计算图像的视差
    输入：左右图像，起始视差，滑动步长，滑动窗口大小
    输出：视差矩阵
    '''
    m, n = img_l.shape
    # 计算等效高斯sigma值
    sigma = (size-1)/8

    # 初始化各数组
    mean_l = np.zeros((m, n))
    mean_r = np.zeros((m, n))
    s = np.zeros((m, n))
    s_l = np.zeros((m, n))
    s_r = np.zeros((m, n))

    # 保存深度信息的数组
    depthmaps = np.zeros((m, n, steps))

    # 计算平均值
    filters.gaussian_filter(img_l, sigma, 0, mean_l)
    filters.gaussian_filter(img_r, sigma, 0, mean_r)

    # 归一化图像
    norm_l = img_l - mean_l
    norm_r = img_r - mean_r

    # 遍历范围内的所有视差值
    for dis in range(steps):
        # 计算NCC
        filters.gaussian_filter(np.roll(norm_l, - dis - start) * norm_r, sigma,
0, s)  # 和归一化
        filters.gaussian_filter(np.roll(norm_l, - dis - start) * np.roll(norm_l,
- dis - start), sigma, 0, s_l)
        filters.gaussian_filter(norm_r * norm_r, sigma, 0, s_r)  # 和反归一化
        depthmaps[:, :, dis] = s / np.sqrt(s_l * s_r)

    # 为每个点选取最佳匹配点并取其视差
    return np.argmax(depthmaps, axis=2) + start

if __name__ == '__main__':
    # 读取图像
    img_l = cv2.imread("tsukuba_l.png",0)
    img_r = cv2.imread("tsukuba_r.png",0)

    # 设置大概的视差范围[start, start+steps-1]
    steps = 15
    start = 3

    # NCC窗口大小
    window_size = 9
```

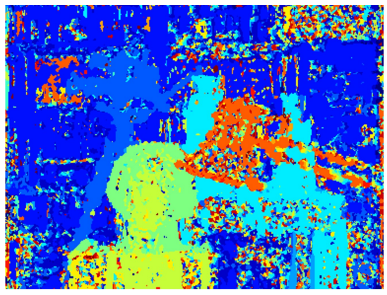
```
    # 计算视差图像
    res = plane_sweep_gauss(img_l, img_r, start, steps, window_size)
    print(res)

    # 绘制伪彩色视差图
    res_plot = np.array((res-start)/(steps-1.0)*255.0, dtype=np.uint8)
    res_plot =  cv2.applyColorMap(res_plot, cv2.COLORMAP_JET)
    cv2.imshow("depth", res_plot)
    cv2.imwrite("depth_wd"+str(window_size)+".png", res_plot)

    cv2.waitKey(0)
```

4 实验结果与分析

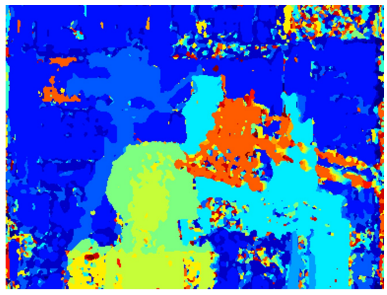
调整参数，将窗口大小分别设为3、9、13、17、25、37，进行实验，得到的结果如下图所示（伪彩色表示视差大小）：



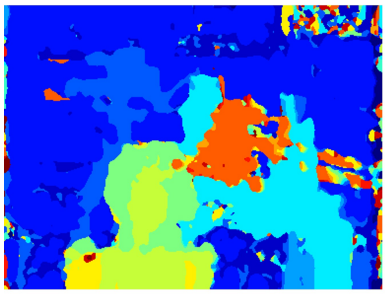
window_size = 3



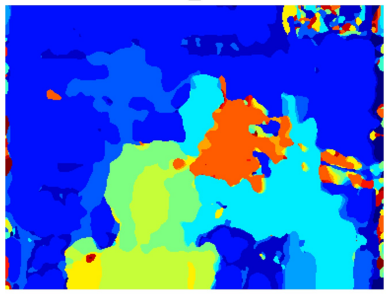
window_size = 9



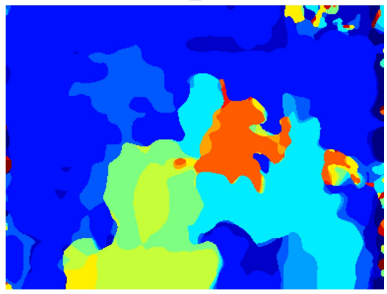
window_size = 13



window_size = 17



window_size = 25



window_size = 37

从这组图片中，我们可以看出，随着窗口大小的增大，图像噪声逐渐减少，但是细节也丢失得越来越多。一些轮廓明显的物体（台灯、雕塑、雕塑后的支架）都是随着窗口增大轮廓更加清晰。而雕塑后面的书桌、书架等细节较多的事物随着窗口增大而丢失细节。其中，窗口大小为9得到的结果轮廓比较明显、细节保留较多，但仍有不少噪声，可滤波后得到相对较好的滤波效果。窗口大小为25得到的结果虽噪声较少，但涂抹感严重，丢失了较多细节，适于大致分析物体的前后关系。

总的来说，使用带高斯加权的归一化相关性(NCC)指标计算的视差图可以较好地去除噪声并分析主要景物的层次关系，但窗口大小不能过大。