三维重建

3D Reconstruction

学　　院	控制科学与工程学院
任课教师	姜伟
学生姓名	申炳琦
学号	22132125

2022 年 12 月 14 日

一、 实验目的与要求

（1） 网站（https://vision.middlebury.edu/stereo/data/）下载任意图片数据集，任选两张立体视觉图像，任选三种以上三维重建方法（可以使用 opencv）实现视差估计，并比较各方法的优缺点。

二、 实验过程与结果

1. 数据集选取

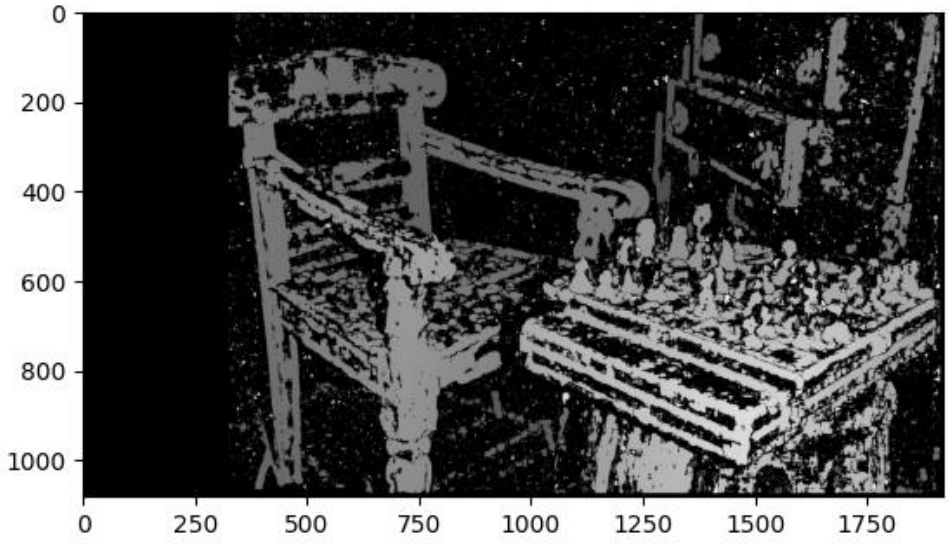
本实验采用了 2021 mobile datasets 中的 chess2 图像来计算其稠密视差，并给出相应的视差图，左右图像如下：

2. 实验结果与对比

本实验基于 OpenCV 开源库，选取 BM、SGBM 和 GC 算法进校对比测试研究。

• BM 算法：

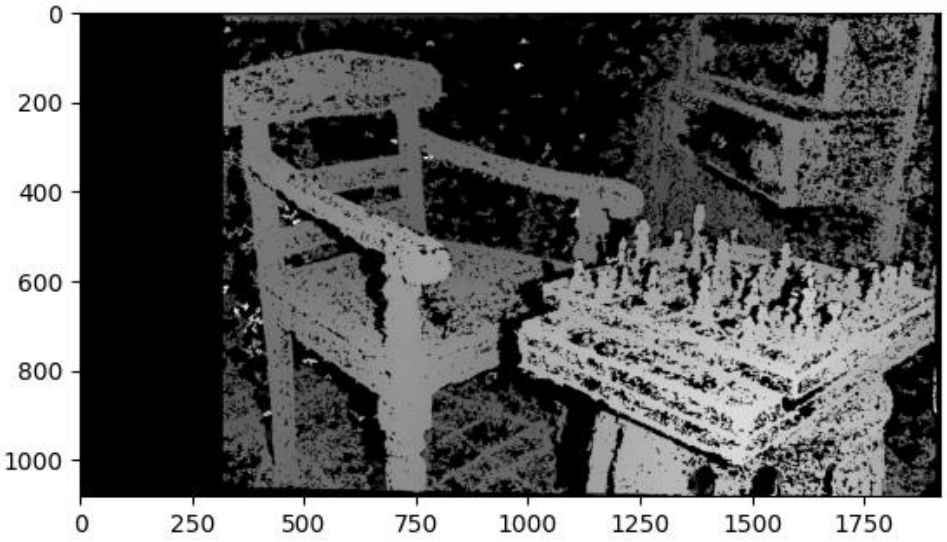
BM 全称是 Block Matching，是一种块匹配算法。其将两张图分成很多的小方块来进行匹配，通过移动方块来匹配另一个图中的方块，发现不同方块在另一个图像中的像素点位置在结合两个摄像头的关系数据（标定的参数中的 translate 和 rotation 矩阵），采用 SAD 方法计算匹配代价，计算出物体的实际深度从而生成相应的深度图，如下图所示。



运行时长：43.85ms

• SGBM 算法：

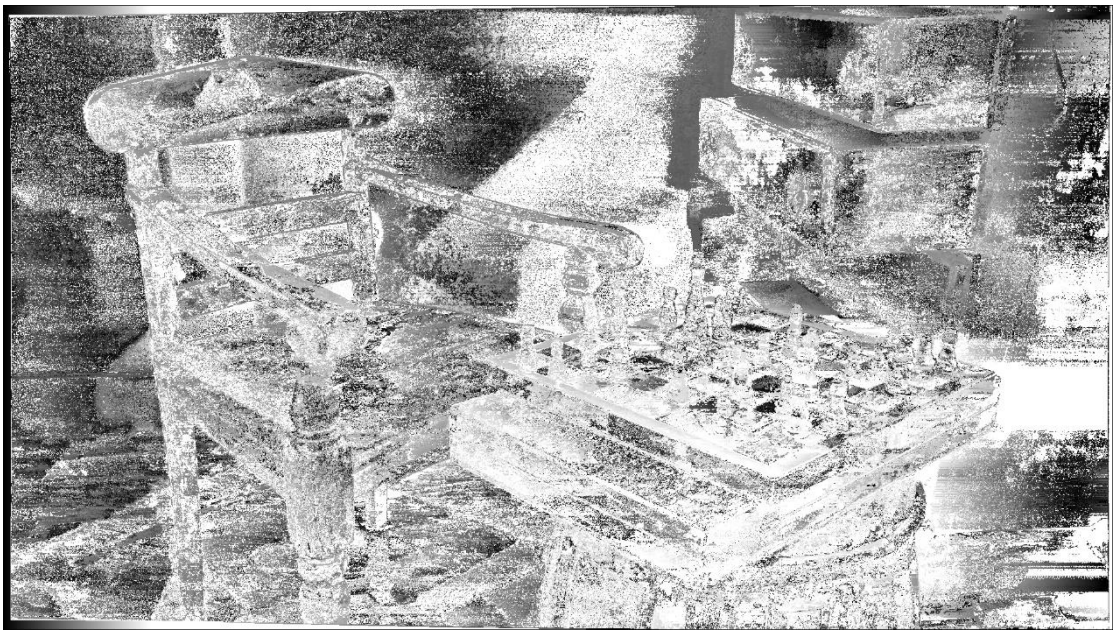
SGBM 全称 Semi-Global Block Matching，是一种全局匹配算法。其立体匹配的效果明显好于局部匹配算法，但是同时复杂度上也要远远大于局部匹配算法。其通过选取每个像素点的 disparity，组成一个 disparity map，设置一个和 disparity map 相关的全局能量函数，使这个能量函数最小化，以达到求解每个像素最优 disparity 的目的。由此得到的深度图如下所示：



运行时长：1153.96ms

• GC 算法：

GC 全称 Graph Cuts，是一种十分有用和流行的能量优化算法，在计算机视觉领域普遍应用于前背景分割（Image segmentation）、立体视觉（stereo vision）、抠图（Image matting）等。



运行时长：503771.25ms

对比上述三种算法，BM 算法的速度最快，但是效果一半；SGBM 算法的效果比 BM 算法好，但同时也更耗时；而 GC 算法速度最慢，但效果却是三者中最好的。将三者的视差效果和处理速度进行对比，对比结果如下：

视察效果：BM < SGBM < GC

处理速度：BM > SGBM > GC

```
#!/usr/bin/env python
import numpy as np
import cv2
from matplotlib import pyplot as plt
import time

def sgbm(imgL, imgR):
    T1 = time.time()
    window_size = 1
    min_disp = 0
    num_disp = 320 - min_disp
    stereo = cv2.StereoSGBM_create(minDisparity =
min_disp,
                                   numDisparities =
num_disp,
                                   blockSize = 5,
                                   P1 = 8 * 3 *
window_size**2,
                                   P2 = 32 * 3 *
window_size**2,
                                   disp12MaxDiff = 1,
                                   uniquenessRatio = 10,
                                   speckleWindowSize = 100,
                                   speckleRange = 32
                                   )

    # disparity = stereo.compute(imgL, imgR)
    disparity = stereo.compute(imgL,
imgR).astype(np.float32) / 16.0

    print((time.time() - T1) * 1000)

    plt.imshow(disparity, 'gray')
    plt.savefig('sgbm.png')
    plt.show()

def bm(imgL, imgR):
    T1 = time.time()
```

```
    # SAD window size should be between 5..255
    block_size = 11

    min_disp = 0
    num_disp = 320 - min_disp
    uniquenessRatio = 10


    stereo = cv2.StereoBM_create(numDisparities =
num_disp, blockSize = block_size)
    stereo.setUniquenessRatio(uniquenessRatio)

    disparity = stereo.compute(imgL,
imgR).astype(np.float32) / 16.0

    print((time.time() - T1) * 1000)

    plt.imshow(disparity,'gray')
    plt.savefig('bm.png')
    plt.show()



if __name__ == "__main__":
    print('loading images...')
    imgL = cv2.imread('chess2/im0.png', 0)
    imgR = cv2.imread('chess2/im1.png', 0)

    bm(imgL, imgR)
    sgbm(imgL, imgR)
```