Computer Vision 2018 Fall Assignment
#2 Disparity Map of Stereo Images

张义飞 — 201821080630
yidadaa@qq.com

UESTC — October 13, 2018

1 双目立体视觉

双目立体视觉的开创性工作始于上世纪的60年代中期。美国MIT的Roberts通过从数字图像中提取立方体、楔形体和棱柱体等简单规则多面体的三维结构，并对物体的形状和空间关系进行描述，把过去的简单二维图像分析推广到了复杂的三维场景，标志着立体视觉技术的诞生。随着研究的深入，研究的范围从边缘、角点等特征的提取，线条、平面、曲面等几何要素的分析，直到对图像明暗、纹理、运动和成像几何等进行分析，并建立起各种数据结构和推理规则。特别是上世纪80年代初，Marr首次将图像处理、心理物理学、神经生理学和临床精神病学的研究成果从信息处理的角度进行概括，创立了视觉计算理论框架。这一基本理论对立体视觉技术的发展产生了极大的推动作用，在这一领域已形成了从图像的获取到最终的三维场景可视表面重构的完整体系，使得立体视觉已成为计算机视觉中一个非常重要的分支。

双目立体视觉（Binocular Stereo Vision）是机器视觉的一种重要形式，它是基于视差原理并利用成像设备从不同的位置获取被测物体的两幅图像，通过计算图像对应点间的位置偏差，来获取物体三维几何信息的方法。

双目立体视觉融合两只眼睛获得的图像并观察它们之间的差别，使我们可以获得明显的深度感，建立特征间的对应关系，将同一空间物理点在不同图像中的映像点对应起来，这个差别，我们称作视差(Disparity)图像。

Question 1

明确本次实验的内容。

1. **目的：**根据双目摄像机获取的两幅图像，计算它们的视差图像（Disparity Image）；
2. **输入：**双目摄像机获取的两幅图像；
3. **输出：**视差图。

2 实验过程

本节中，我们首先调用Opencv自带的SGBM算法计算原始双目图像的视差图，然后调研SGBM算法的几个参数对最终效果的影响。

2.1 计算视差图

原始图像和计算出的视差图分别如图1，2，3所示。

2.2 将灰度图转化为伪彩色图像

为了使生成的图像便于观察，使用opencv的applyMapColor函数将灰度图转化为彩色图像。

Figure 1: 左目图像

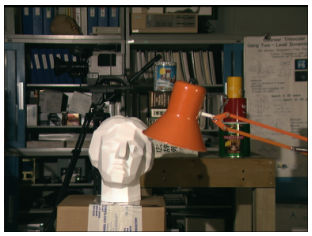


Figure 2: 右目图像

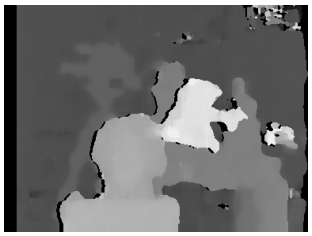
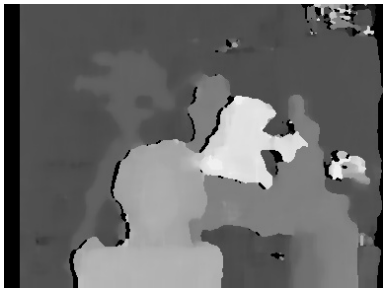


Figure 3: 视差图



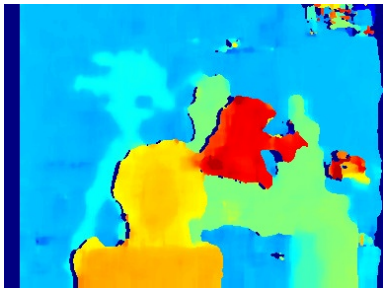


Figure 4: 将灰度图映射为伪彩色图像

2.3 观察SGBM算法的不同参数对结果的影响

SGBM重要参数如下所示：

1. **MinDisparity**，即最小视差，本实验中将其设置为0，因为两个摄像头是前向平行放置，相同的物体在左图中一定比在右图中偏右。如果为了追求更大的双目重合区域而将两个摄像头向内偏转的话，这个参数是需要考虑的。

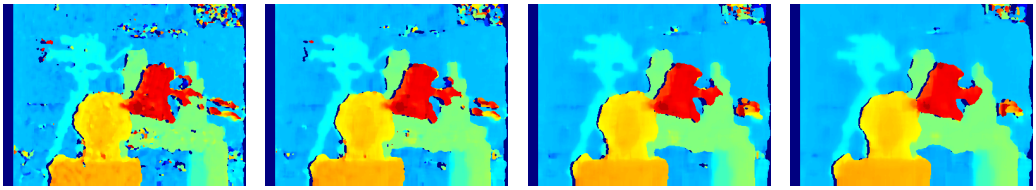
2. **UniquenessRatio**，主要可以防止误匹配，此参数对于最后的匹配结果是有很大的影响。立体匹配中，宁愿区域无法匹配，也不要误匹配。如果有误匹配的话，碰到障碍检测这种应用，就会很麻烦。该参数不能为负值，一般5-15左右的值比较合适，int型。

3. **BlockSize**，SAD窗口大小，容许范围是[5,255]，一般应该在 5x5..21x21 之间，参数必须为奇数值, int型。

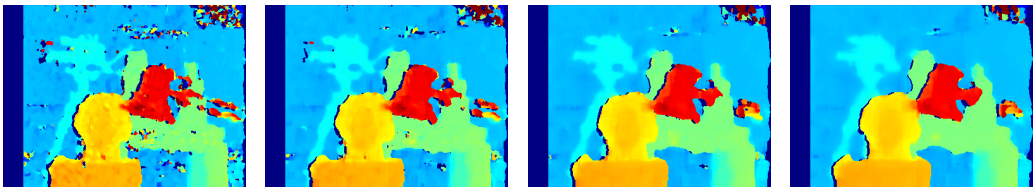
4. **NumDisparities**，视差窗口，即最大视差值与最小视差值之差,窗口大小必须是 16的整数倍，int型。

在SGBM算法的参数中，对视差生成效果影响较大的主要参数是BlockSize、NumDisparities和UniquenessRatio三个，一般只需对这三个参数进行调整，其余参数按默认设置即可。本节将着重观察BlockSize和NumDisparities两个参数对结果的影响。图5展示了不同的参数对最终生成的视差图的影响。

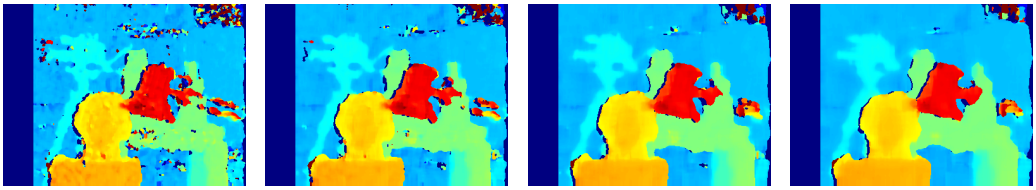
可以看到，较大的BlockSize可以使得生成的视差图更加平滑，而较大的NumDisparities会使画面左侧出现黑色边框，所以使用较小的NumDisparities和较大的BlockSize可以获得比较好的视差图效果。



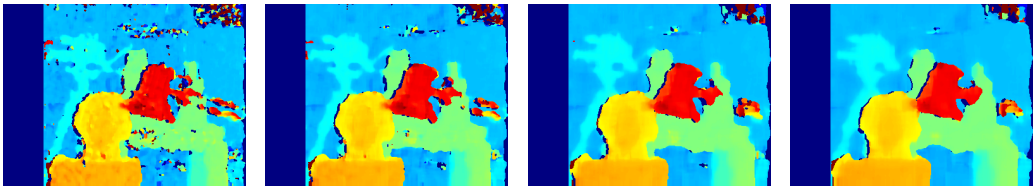
(a) $N_d = 16, S_b = 7$ (b) $N_d = 16, S_b = 11$ (c) $N_d = 16, S_b = 15$ (d) $N_d = 16, S_b = 19$



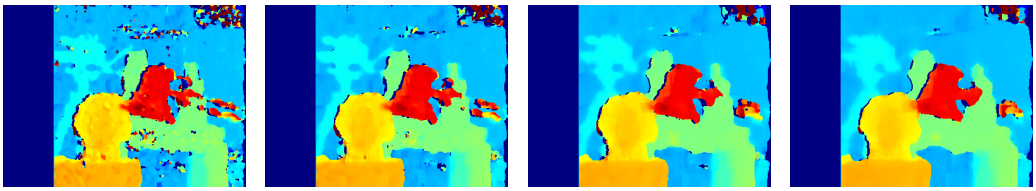
(e) $N_d = 32, S_b = 7$ (f) $N_d = 32, S_b = 11$ (g) $N_d = 32, S_b = 15$ (h) $N_d = 32, S_b = 19$



(i) $N_d = 48, S_b = 7$ (j) $N_d = 48, S_b = 11$ (k) $N_d = 48, S_b = 15$ (l) $N_d = 48, S_b = 19$



(m) $N_d = 64, S_b = 7$ (n) $N_d = 64, S_b = 11$ (o) $N_d = 64, S_b = 15$ (p) $N_d = 64, S_b = 19$



(q) $N_d = 80, S_b = 7$ (r) $N_d = 80, S_b = 11$ (s) $N_d = 80, S_b = 15$ (t) $N_d = 80, S_b = 19$

Figure 5: 不同参数对结果的影响

3 附录

本实验代码已在Github开源，请访问：https://github.com/Yidadaa/Computer-Vision-Assignment/tree/master/Assignment-2。

```
import os
import numpy as np
from matplotlib import pyplot as plt

class DisparityMap:
    def __init__(self, numDisparities, blockSize):
        self.stereo = cv2.StereoSGBM_create(0, numDisparities, blockSize)

    def computeDsiparity(self, l, r):
        return self.stereo.compute(l, r)

    def saveToFile(self, filename, img):
        cv2.imwrite(filename, img)

    def setParams(self, numDisparities, blockSize):
        self.stereo = cv2.StereoSGBM_create(0, numDisparities, blockSize)

if __name__ == '__main__':
    instance = DisparityMap(16, 15)
    lImage = cv2.imread('./dataset/l.png', 0)
    rImage = cv2.imread('./dataset/r.png', 0)

    result_dir = './result'
    for f in os.listdir(result_dir):
        os.remove(result_dir + '/' + f)

    example_gray = 'result/example_gray.jpg'
    example_fakergb = 'result/example_fakergb.jpg'
    disparity = instance.computeDsiparity(lImage, rImage)
    instance.saveToFile(example_gray, disparity)
    gray = cv2.imread(example_gray)
    fakergb = cv2.applyColorMap(gray, cv2.COLORMAP_JET)
    instance.saveToFile(example_fakergb, fakergb)

    numDisparities = list(range(16, 16 * 6, 16))
    blockSize = list(range(7, 23, 4))
    for n in numDisparities:
        for b in blockSize:
            instance.setParams(n, b)
            disparity = instance.computeDsiparity(lImage, rImage)
            fname = 'result/%s_%s.png' % (n, b)
            instance.saveToFile(fname, disparity)
            disparity = cv2.imread(fname, 0)
            disparity = cv2.applyColorMap(disparity, cv2.COLORMAP_JET)
            instance.saveToFile(fname, disparity)
```