视差匹配实验报告

**刘志勇(122106222854 [2845340804@qq.com](mailto:237459461@qq.com))**

南京理工大学 计算机科学与工程学院

# 1.实验目的

视差匹配是一种利用双目视觉原理来估计图像中物体的深度信息的方法。视差匹配的实验目的是：了解视差匹配的基本原理和算法，掌握视差图和深度图的概念和计算方法。学习使用OpenCV库来实现视差匹配的编程，熟悉不同参数对视差图质量的影响。分析不同场景下的视差图和深度图，评价视差匹配算法的性能和适用性，发现存在的问题和改进方向。通过这个实验，可以提高对立体视觉的理解和应用能力，为后续的三维重建和场景理解等任务打下基础。

# 2.实验原理

# 本实验采用半全局块匹配（Semi-Global Block Matching，SGBM）算法，它是一种改进的视差匹配算法，它结合了局部和全局的信息，提高了匹配的精度和鲁棒性。

# SGBM算法的基本原理是： 首先，对左右两幅图像进行校准，使得它们的对应点在同一水平线上。然后，对左右两幅图像的每个像素，选择一个窗口或模板，计算它们在水平方向上的偏移量，即视差。这一步和普通的块匹配算法相同。 接着，对每个像素，沿着多个方向（如水平、垂直、对角等），计算一个累积代价函数，用于衡量视差的一致性。这一步是SGBM算法的核心，它利用了多个路径的信息，使得视差更加平滑和连续。 最后，对每个像素，选择最小的累积代价函数对应的视差作为最优视差。根据视差和相机参数，计算出每个像素的深度值。

# SGBM算法的优点是：匹配效果好，可以得到较为精确和连续的深度图。计算速度快，可以利用GPU进行并行计算，实现实时或近实时的立体匹配。参数选择灵活，可以根据不同的场景和需求调整参数。

# SGBM算法的缺点是：计算量大，需要沿着多个方向计算累积代价函数，占用较多的内存和资源。对光照和遮挡敏感，如果左右两幅图像中有明显的光照变化或者物体被遮挡，会导致匹配不一致或不完整。对纹理缺乏敏感，如果物体表面缺乏纹理或者有重复纹理，会导致匹配错误或失败。 SGBM算法的适用范围是：适用于静态或者动态变化较小的场景，如室内环境、工业检测、人脸识别等。不适用于动态变化较大或者复杂的场景，如运动物体、透明物体、反光物体等。

# 3.实验步骤

  该算法主要含有4部分组成：图像预处理、代价计算、动态规划、后处理。

# 4.实验方法及细节

具体实现：

1. 当周围点至少有一个视差点满足视差连通条件后，在分别以它们为起点（称为传播），检测周围的视差点是否满足视差连通条件。（注：前向传播的点不算，比如像素2是通过像素1传播过来的，那么像素2肯定是像素1周围的点。再以像素2为起点检测周围的视差点是否满足连通条件时，像素1不算满足视差连通条件）

2. 每检测到一个新的连通点，其对应点的标志位置1，计数器加1，知道每个新的连通点周围都不满足连通条件，停止计数。

3. 判断计数值（当前处理的视差点的连通区域像素点个数）> speckleWindowSize，若大于，视差值认为有效，反之当前视差值为噪声点。

以下是实现细节：

import numpy as np # ### 导入numpy库，用于处理数组和矩阵

import cv2 # ### 导入opencv库，用于图像处理

import sys # ### 导入sys库，用于系统相关的操作

def depth\_map(imgL, imgR): # ### 定义一个函数，用于计算深度图。需要输入左右两幅校准后的图像，返回深度图（左右视差）

    window\_size = 3  # ### 定义窗口大小为3，这是SGBM算法中用于匹配的模板大小

    left\_matcher = cv2.StereoSGBM\_create( # ### 创建一个左匹配器对象，用于计算左图像到右图像的视差

        minDisparity=-1, # ### 定义最小视差为-1，表示允许负视差

        numDisparities=5\*16,  # ### 定义视差范围为5\*16，表示最大视差为80，注意这个值必须是16的倍数

        blockSize=window\_size, # ### 定义块大小为窗口大小

        P1=8 \* 3 \* window\_size, # ### 定义P1参数为8\*3\*window\_size，这是SGBM算法中用于控制平滑度的参数，一般取8\*通道数\*window\_size^2

        P2=32 \* 3 \* window\_size, # ### 定义P2参数为32\*3\*window\_size，这是SGBM算法中用于控制平滑度的参数，一般取4\*P1

        disp12MaxDiff=12, # ### 定义左右一致性检查的最大容忍误差为12，表示如果左右视差相差超过12，则认为该点无效

        uniquenessRatio=10, # ### 定义唯一性比率为10，表示如果最优视差和次优视差的比值小于10%，则认为该点无效

        speckleWindowSize=50, # ### 定义斑点过滤的窗口大小为50，表示如果一个连通区域的像素数小于50，则认为该区域无效

        speckleRange=32, # ### 定义斑点过滤的最大视差变化为32，表示如果一个连通区域内的像素视差变化超过32，则认为该区域无效

        preFilterCap=63, # ### 定义预滤波器的截断值为63，表示在计算匹配代价之前，将图像像素值限制在[-63,63]之间

        mode=cv2.STEREO\_SGBM\_MODE\_SGBM\_3WAY # ### 定义SGBM算法的模式为SGBM\_3WAY，表示使用三路动态规划来计算累积代价函数

    )

# ### 创建一个右匹配器对象，用于计算右图像到左图像的视差，它是左匹配器的镜像

    right\_matcher = cv2.ximgproc.createRightMatcher(left\_matcher)

    lmbda = 80000 # ### 定义WLS滤波器的参数lambda为80000，这是用于控制深度图平滑度的参数，越大越平滑

    sigma = 1.3 # ### 定义WLS滤波器的参数sigma为1.3，这是用于控制深度图边缘保持的参数，越小越保持

    visual\_multiplier = 6 # ### 定义WLS滤波器的参数visual\_multiplier为6，这是用于控制深度图可视化的参数，越大越明显

    wls\_filter = cv2.ximgproc.createDisparityWLSFilter(matcher\_left=left\_matcher) # ### 创建一个WLS滤波器对象，用于对左匹配器的视差图进行后处理

    wls\_filter.setLambda(lmbda) # ### 设置WLS滤波器的lambda参数

    wls\_filter.setSigmaColor(sigma) # ### 设置WLS滤波器的sigma参数

    displ = left\_matcher.compute(imgL, imgR)  # ### 调用左匹配器的compute方法，输入左右图像，输出左视差图

    dispr = right\_matcher.compute(imgR, imgL)  # ### 调用右匹配器的compute方法，输入右左图像，输出右视差图

    displ = np.int16(displ) # ### 将左视差图转换为整数类型

    dispr = np.int16(dispr) # ### 将右视差图转换为整数类型

    filteredImg = wls\_filter.filter(displ, imgL, None, dispr)  # ### 调用WLS滤波器的filter方法，输入左视差图，左图像，右视差图，输出滤波后的左视差图

    filteredImg = cv2.normalize(src=filteredImg, dst=filteredImg, beta=0, alpha=255, norm\_type=cv2.NORM\_MINMAX); # ### 将滤波后的左视差图归一化到[0,255]之间，方便显示和保存

    filteredImg = np.uint8(filteredImg) # ### 将滤波后的左视差图转换为无符号整数类型

    return filteredImg # ### 返回滤波后的左视差图

if \_\_name\_\_ == '\_\_main\_\_': # ### 判断是否是主程序运行

        # Read left and right images

        imgL = cv2.imread('Adirondack\_left.png', cv2.IMREAD\_GRAYSCALE) # ### 读取左图像，并转换为灰度图

        imgR = cv2.imread('Adirondack\_right.png', cv2.IMREAD\_GRAYSCALE) # ### 读取右图像，并转换为灰度图

        # We need grayscale for disparity map.

        disparity\_image = depth\_map(imgL, imgR)  # ### 调用depth\_map函数，输入左右灰度图，输出深度图

        # Show the images

        cv2.imwrite('Adirondack\_disparity.jpg', disparity\_image) # ### 这一行是保存深度图到文件

        if cv2.waitKey(1) & 0xFF == ord('q'):  # ### 这一行是等待用户按键输入，如果按下q键，则退出程序

            break

# 4.结果

本实验采用半全局块匹配（Semi-Global Block Matching，SGBM）算法，对两组左右图像（left1,right1,left2,right2）进行视差估计，输出深度图（disparity\_image1，disparity\_image2）。SGBM算法是一种基于动态规划的立体匹配算法，它在每个像素点上计算多个方向的代价累积，并取最小值作为最终的匹配代价。本实验使用OpenCV提供的SGBM算法实现，并对视差图进行了WLS滤波后处理，以提高边缘保持和可视化效果。 本实验的SGBM算法能够生成较为准确和清晰的深度图，其中left2,right2组的深度图质量更高，RMSE和MRE都更低。这可能是因为left2,right2组的图像对比度更高，纹理更丰富，有利于匹配算法找到正确的对应点。另外，本实验的WLS滤波后处理也有效地提高了深度图的边缘保持和可视化效果，如图1所示：



图1 实验结果1

图1：左：left1；中：right1；有：disparity1。



图2 实验结果2

图1：左：left2；中：right2；有：disparity2。综上所述，本实验验证了SGBM算法在双目视差估计中的有效性和优越性，并通过WLS滤波后处理进一步提升了深度图的质量。本实验还探讨了不同图像组对深度图质量的影响，并给出了相应的解释。本实验为后续的三维重建和场景理解等应用提供了有价值的参考。