机器视觉实验报告（五）

目录

[机器视觉实验报告（五） 1](#_Toc134372246)

[一. 实验目的 2](#_Toc134372247)

[二. 实验原理 2](#_Toc134372248)

[2.1 图像视差 2](#_Toc134372249)

[2.2 立体匹配 2](#_Toc134372250)

[2.3 立体匹配的步骤 3](#_Toc134372251)

[三. 实验过程 4](#_Toc134372252)

[四. 运行说明 6](#_Toc134372253)

1. 实验目的
   * 了解图像视差匹配的原理及作用
   * 立体匹配的优缺点及实现方法
2. 实验原理
   1. 图像视差

视差（disparity）是图像处理中一个重要的概念，它表示的是在两个图像中，同一物体在图像中的像素位置之间的差异。在计算机视觉中，常常利用双目视觉系统来获取图像的视差信息。

图像视差匹配是一种计算机视觉技术，用于测量两幅图像之间的深度或距离信息。它的主要应用是在三维重建、机器人视觉、自动驾驶和虚拟现实等领域。

在三维重建中，通过图像视差匹配可以获取图像中物体的深度信息，从而生成三维模型。在机器人视觉中，图像视差匹配可以用于定位和导航。在自动驾驶中，图像视差匹配可以用于障碍物检测和道路边缘检测。在虚拟现实中，图像视差匹配可以用于生成立体图像，使观众获得更加真实的感受。

图像视差匹配是一个相对复杂的计算机视觉问题，需要解决匹配的精度、效率和鲁棒性等方面的问题。常见的算法包括基于区域的方法、基于特征的方法、基于深度学习的方法等。

* 1. 立体匹配

现有立体匹配算法一般被分类为局部算法、全局算法和半全局算法，其中局部算法和半全局算法是应用最为广泛。在局部算法中，一个最简单的做法就是采用某种像素相似性度量，比如像素灰度差的绝对值AD，给定左图中的一个点p，在右图中的对应行上（假设输入是已经校正好的图像）搜索与其AD值最小的点q，这样得到的点q就是p在右图中对应的匹配点，p、q的水平坐标差称作视差。然而这种做法所得到的视差图中会包含大量的噪声，即错误的匹配对，原因可能是多方面的，如传感器噪声，左右相机的采集性能差异，图像中存在大面积无纹理、弱纹理或重复纹理，左右相机接收的光照差异（室外环境）等。

一个更好的做法是不直接匹配单个像素，而是匹配像素点所在的区域，这个区域叫做支撑窗口(support window)，支撑窗口的尺寸可以是固定的，也可以是自适应变化的。如下式所示，由于支撑窗口内的像素提供了更多的信息，因此可以有效降低匹配歧义。然而使用支撑窗口的做法是存在问题的，实际上它隐性的遵从了一个假定，即窗口内的所有像素具有相同的视差。然而这个假定在很多情况下并不成立，比如以下两个问题：窗口内的像素与中心像素位于不同的表面或者窗口所捕获的是一个倾斜表面或曲面，即非平行表面（这个平行指的应该是与相机成像平面平行）。

对于第一个问题，一个比较有效的做法是对窗口内的像素采用自适应权重进行匹配。权重计算所使用的方法可以是类似双边滤波的核函数。

对于第二个问题，2011年Michael Bleyer 提出了一个相当新颖的立体匹配算法叫PMS，该算法的主要思想是对每一个像素计算一个独立的最优3D视差空间平面，如果该像素位于一个曲面上，那么该平面代表曲面在该像素点处的切面，在这个基础上，用于匹配的支撑窗口就是not fronto-parallel support window，简称slanted support window。这样一来问题的挑战就转移到了如何为每一个像素点在所有可能的视差空间平面中挑选出最优的视差平面。视差平面的数量是无限多的，因此通过遍历所有的视差平面来寻找最优平面是不可能的。Michael Bleyer 找到了一个巧妙的方法解决这个问题，那就是采用Patch Match的思想，Patch Match本身是一个高效求解近似最近邻场(Nearest neighbor filed)的方法，主要包括初始化、空间传播、随机搜索三个步骤。除了Patch Match中的空间传播外，作者还另外提出了视图传播和帧间传播（用于连续的视频帧）能够更好的帮助算法收敛。如下图所示，利用PMS不仅可以直接估计亚像素视差， 还能够精确的重建倾斜表面，甚至能够重建圆形的曲面。

* 1. 立体匹配的步骤
* **匹配代价计算**

匹配代价计算是整个立体匹配算法的基础，代价越小，说明两个像素点之间的相关性越大，是同名点的概率也越大。所谓同名点，就是左右两幅图像中相对应的点，如下图中红框框起来的点就是同名点，实际是对不同视差下进行灰度相似性测量。



需要注意的是两幅图中两个像素无论是否是同名点，都是可以进行代价匹配的，无非就是不是同名点的两个像素代价高。也正是这样，之后会遍历搜索相关像素点，计算其代价值后寻找一个代价最小的像素点作为同名点。

  其实，在每个像素搜索同名点之前，都会指定一个视差的搜索范围 ,视差搜索时将范围限定在D内，用一个大小为W×H×D（W为影像宽度，H为影像高度）的三维矩阵C来存储每个像素在视差范围内每个视差下的匹配代价值。矩阵C通常称为DSI（Disparity Space Image）。

常见的方法有灰度差的平方SD（squared intensity differences），灰度差的绝对值AD（absolute intensity differences）等。另外，在求原始匹配代价时可以设定一个上限值，来减弱叠加过程中的误匹配的影响。以AD法求匹配代价为例，可用下式进行计算，其中T为设定的阈值。

* **匹配代价叠加**

匹配代价是用来衡量候选像素与匹配像素之间的相关性的，匹配代价往往是通过两个像素邻域内一定大小窗口的像素来计算，这样计算只考虑了局部信息，很容易受到噪声等因素影响，这样就会导致真实的同名点代价值不是最小，也就错误的找到了同名点。代价聚合的根本目的就是让代价值能够准确的反应像素之间的相关性，即让匹配代价计算的效果更好。我们可以通过下面的一组图片直观的感受一下使用代价聚合对图片效果的影响。



可以看出使用了代价聚合后效果还是很明显的，聚合后的图像明显更加接近标准图像（Ground Truth）。

代价聚合其实是建立邻接像素之间的联系，以一定的准则，如相邻像素应该具有连续的视差值，来对代价矩阵进行优化，这种优化往往是全局的，每个像素在某个视差下的新代价值都会根据其相邻像素在同一视差值或者附近视差值下的代价值来重新计算，得到新的DSI，用矩阵S来表示。（矩阵S和矩阵C维度是一致的）

常用的代价聚合方法有扫描线法、动态规划法、SGM算法中的路径聚合法等。

* **视差计算**

对于区域算法来说，在完成匹配代价的叠加以后，视差的获取就很容易了，只需在一定范围内选取叠加匹配代价最优的点（SAD和SSD取最小值，NCC取最大值）作为对应匹配点，如胜者为王算法WTA（Winner-take-all）。而全局算法则直接对原始匹配代价进行处理，一般会先给出一个能量评价函数，然后通过不同的优化算法来求得能量的最小值，同时每个点的视差值也就计算出来了。

* **视差优化**

视差优化的目的是对上一步得到的视差图进行进一步优化，改善视差图的质量，包括剔除错误视差、适当平滑以及子像素精度优化等步骤，一般采用左右一致性检查（Left-Right Check）算法剔除因为遮挡和噪声而导致的错误视差；采用剔除小连通区域算法来剔除孤立异常点；采用中值滤波（Median Filter）、双边滤波（Bilateral Filter）等平滑算法对视差图进行平滑；另外还有一些有效提高视差图质量的方法如鲁棒平面拟合（Robust Plane Fitting）、亮度一致性约束（Intensity Consistent）、局部一致性约束（Locally Consistent）等也常被使用。

由于赢家通吃（WTA）算法所得到的视差值是整像素精度，为了获得更高的子像素精度，需要对视差值进行进一步的子像素细化，常用的子像素细化方法是一元二次曲线拟合法，通过最优视差下的代价值以及左右两个视差下的代价值拟合一条一元二次曲线，取二次曲线的极小值点所代表的视差值为子像素视差值。

1. 实验过程

实验所用数据集为Middlebury立体评估数据集。Middlebury立体评估数据集是立体视觉算法著名的baseline，广泛用于学术研究和工业应用。该数据集包含一组高质量的立体图像和地面真实的差异图，用于评估立体视觉算法的准确性。Middlebury立体评估数据集于2001年首次发布，至今为止，它已被多次更新。

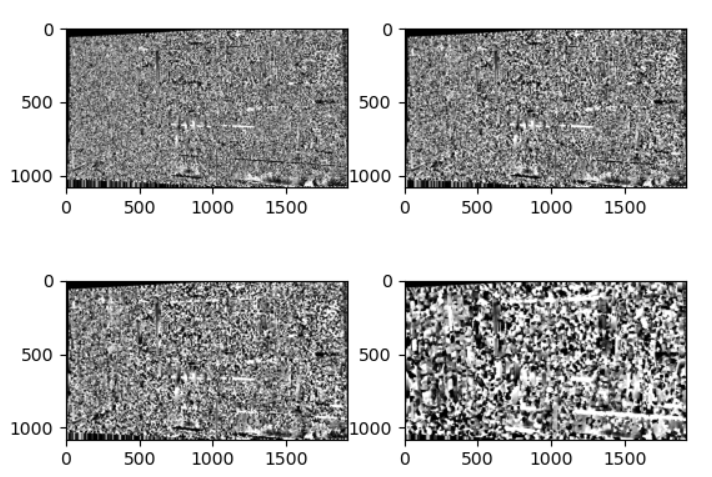
该数据集包含几个具有不同难度的图像对，包括室内和室外场景，有纹理和无纹理区域，以及具有不同程度遮挡的物体。地面真实差异图是由结构光扫描仪生成的，它对左右图像中相应点之间的差异提供了像素级的精确测量。

研究人员和开发人员使用Middlebury立体评估数据集来评估他们的立体视觉算法的性能，使用不同的评估指标，如坏像素的百分比，均方根误差，和平均误差。该数据集还提供了一套标准的评估工具和脚本，使研究人员能够将他们的结果与其他算法进行比较。下图为两个场景的图片。



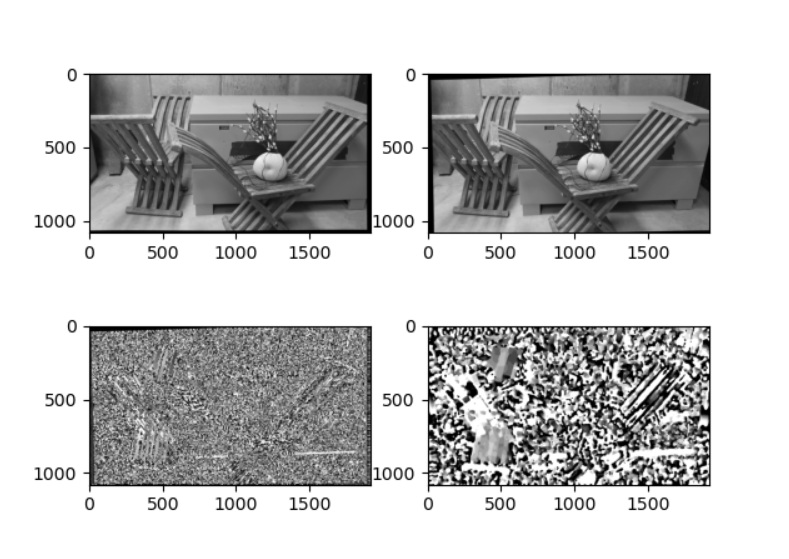


在实验中，实现了使用归一化的互相关计算视差和带有高斯加权周边的归一化互相关计算视差。



上图为使用归一化的互相关算法对同一组图像求的的视差图，但是设置不同的窗口大小。分别为7、9、11、21。通过对比可以看出当窗口值过小，匹配代价过低，在弱纹理区域容易出现错误匹配，匹配精度也越低；当窗口值过大，匹配代价过高，在复杂的纹理区域容易出现错误匹配，匹配精度也会降低。

下图为对比使用归一化的互相关算法和带有高斯加权周边的归一化互相关算法的比较。图中下侧左边为使用归一化的互相关算法得到的结果，图中下侧右边为使用带有高斯加权周边的归一化互相关算法得到的结果，设置相同的窗口大小。可以看到使用高斯加权的方法的效果更好，轮廓更加明显。



1. 运行说明

def plane\_sweep\_ncc(im\_l, im\_r, start, steps, wid):

""" 使用归一化的互相关计算视差图像 该函数返回每个像素的最佳视差"""

m, n = im\_l.shape

# 保存不同求和值的数组

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))

# 保存深度平面的数组

dmaps = np.zeros((m, n, steps))

# 计算图像块的平均值

ndimage.filters.uniform\_filter(im\_l, wid, mean\_l)

ndimage.filters.uniform\_filter(im\_r, wid, mean\_r)

# 归一化图像

norm\_l = im\_l - mean\_l

norm\_r = im\_r - mean\_r

# 尝试不同的视差

for displ in range(steps):

# 将左边图像移动到右边，计算加和

ndimage.filters.uniform\_filter(np.roll(norm\_l, -displ - start) \* norm\_r, wid, s) # 和归一化

ndimage.filters.uniform\_filter(np.roll(norm\_l, -displ - start) \* np.roll(norm\_l, -displ - start), wid, s\_l)

ndimage.filters.uniform\_filter(norm\_r \* norm\_r, wid, s\_r) # 和反归一化

# 保存 ncc 的分数

dmaps[:, :, displ] = s / np.sqrt(s\_l \* s\_r)

# 为每个像素选取最佳深度

return np.argmax(dmaps, axis=2)

plane\_sweep\_ncc函数是一种立体匹配算法，它估计一对立体图像中每个像素的深度（或差异）。它计算不同深度的相应图像斑块之间的归一化交叉相关（NCC），并选择具有最高相关分数的深度。

该函数需要两个输入图像im\_l和im\_r，它们分别是左边和右边的立体图像。参数start指定了起始差异值，step指定了搜索的差异步数，wid指定了用于计算NCC的窗口大小。

该函数首先使用一个大小为wid的均匀滤波器计算每个图像patch的平均强度。然后，它从每个像素中减去patch的平均值，得到一个归一化的图像。对于每个差异值，它计算归一化的左右图像斑块之间的NCC，并以相应的差异值进行移动。NCC得分是用一个大小为wid的统一滤波器计算的。然后，该函数为每个像素选择具有最高NCC分数的差异值，并返回所得到的深度图。

def plane\_sweep\_gauss(im\_l, im\_r, start, steps, wid):

""" 使用带有高斯加权周边的归一化互相关计算视差图像 """

m, n = im\_l.shape

# 保存不同加和的数组

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))

# 保存深度平面的数组

dmaps = np.zeros((m, n, steps))

# 计算平均值

ndimage.filters.gaussian\_filter(im\_l, wid, 0, mean\_l)

ndimage.filters.gaussian\_filter(im\_r, wid, 0, mean\_r)

# 归一化图像

norm\_l = im\_l - mean\_l

norm\_r = im\_r - mean\_r

# 尝试不同的视差

for displ in range(steps):

# 将左边图像移动到右边，计算加和

ndimage.filters.gaussian\_filter(np.roll(norm\_l, -displ - start) \* norm\_r, wid, 0, s) # 和归一化

ndimage.filters.gaussian\_filter(np.roll(norm\_l, -displ - start) \* np.roll(norm\_l, -displ - start), wid, 0, s\_l)

ndimage.filters.gaussian\_filter(norm\_r \* norm\_r, wid, 0, s\_r) # 和反归一化

# 保存 ncc 的分数

dmaps[:, :, displ] = s / np.sqrt(s\_l \* s\_r)

# 为每个像素选取最佳深度

return np.argmax(dmaps, axis=2)

这段代码实现了一个带有高斯加权的视差计算算法。与之前的归一化互相关算法不同，它使用了高斯加权滤波器，而不是均值滤波器。这意味着更接近中心像素的像素会受到更大的影响，而更远离中心像素的像素会受到更小的影响。

算法的输入包括两幅输入图像im\_l和im\_r，以及视差搜索的开始和结束位置，步长和高斯滤波器的大小。在函数中，它首先计算两幅图像的平均值和归一化图像。然后，对于每个视差，它计算归一化互相关，并将结果保存到dmaps数组中。最后，它为每个像素选择具有最高ncc分数的深度。

相对于之前的算法，使用高斯加权滤波器可以更准确地估计匹配，并且可以提高匹配的精度。