## 基于匹配代价卷融合和多步插值策略的立体匹配

## 1.摘要

在这篇论文里，我们提出一种局部立体匹配方法，它采用了新的视差计算策略和一种联合插值视差优化策略。初始代价通过组合绝对颜色差和census变换来获得。用引导滤波进行代价聚合。不同于传统的WTA策略，一种结合邻域信息的视差计算策略被提出来提高从代价卷中选择视差的正确率。在视差优化中，联合两种插值策略进行错误点的视差填充。利用Middleburry平台的图片集进行多组对比实验，证明了本文算法的有效性。

## 引言

立体匹配是计算机视觉领域的研究热点，在许多应用场景，如虚拟现实，机器人，自动驾驶中发挥着重要作用【a】。它通过匹配立体图像对的像素点来获取视差图。根据【b】，立体匹配算法可分为全局匹配和局部匹配。全局算法将视差计算转化为能量最小化问题，可以通过置信度传播【c】、图割【d】等算法来求解，匹配精度高，但计算耗时；局部算法通过构建局部支持窗口来降低匹配的模糊性，实现简单，运行速度快，一些局部算法的精度可以媲美全局算法。局部立体匹配可分为匹配代价计算、代价聚合、视差计算、视差优化四个步骤。

匹配代价计算近些年的创新集中在对已有方法的融合上，如AD-census【ADcensus】、AD-gradient【guideFilter】、Census-MI-gradient【e】，比起单个方法，融合方法在精度上得到提升，原因是它们利用了多个维度的信息而当方法只利用一个维度的信息。代价聚合作为局部匹配算法最关键的步骤，是研究者最关注的点，其面临两个问题：如何使得支持像素点尽可能多；如何保证支持像素点来自目标像素点的相同或相近视差层，这两个问题存在矛盾性，支持像素点越多，来自不同视差层的可能性就越大。针对这一矛盾，有两个解决思路，一是采用自适应加权平均法，如【ASW】，它根据一定规则计算支持点和目标点的相关程度，相关度越大，赋的权重越大，在使用大窗口的情况下，也能限制非相同视差层点的干扰，但是，【ASW】作为一种滤波方法，其时间复杂度和和窗口大小成正比，所以另一种和窗口大小无关的保边滤波后-引导滤波被He提出后，其很快就被应用到代价聚合中来；二是采取自适应窗口法，最经典的就是基于十字交叉的自适应窗口法【Cross-Base】，和【ASW】相同的思路，它也把颜色和距离作为了邻近像素和目标像素相关程度的度量，并以此构建和物体边缘贴合的窗口。大量的研究力量被用在代价聚合和代价计算上，对视差优化的研究相对较少，并且很少有针对视差计算的改进。

在这篇论文中，我们提出了两个策略来进一步提高立体匹配的精度。首先，一种结合邻域的视差计算策略被提出以提升初始视差图的精度，对比WTA策略，使重复纹理区的误匹配明显降低；第二，我们提出一种背景插值和区域投票插值相结合的联合插值策略，对比单个插值策略，联合插值取得了更好的表现。为验证上述两个策略的有效性，我们对比了在前两个步骤——代价计算和代价聚合分别采用多种算法时，被提策略和原来策略的表现，实验结果表明，所提策略在不同的前步骤算法环境中都能取得更好的效果。

## 2.本文算法

2.1 代价计算

介绍AD-Census、AD-梯度

2.2 代价聚合

介绍基于十字交叉的代价聚合、引导滤波

2.3 结合邻域的视差计算

WTA(Winner-Takes-All)策略通常被用来进行视差计算，其公式如下：

(11)

式中，为目标像素，为视差，为代价值，为视差，为设定的最大视差。

该策略操作简单，执行速度快，能够求得最小代价对应的视差。但是，在弱纹理和重复纹理区，会出现正确视差的代价不是最小且存在多个接近的小代价值的现象；另外，噪声也会使最小代价值和正确视差不对应。WTA策略不能正确处理上述两种情况。

针对上述问题，我们利用邻域信息来降低视差选择的歧义性。首先，我们从匹配代价卷构建每个点的候选视差集，若候选视差集中的元素差距过大，说明该点的视差存在较大不确定性，需要借助邻域已经确定的视差来辅助该点的视差选择，即从候选视差中选择和已确定的邻域视差最接近的视差；假定多数情况下邻近区域的视差是相近的，当候选视差集中存在视差接近的元素时，我们认为正确视差存在于这些视差中，而距离这些接近的视差较远的单个元素，则判定是误差，将其剔除，当邻域确定视差和目标点的候选视差集接近时，引入邻域的视差信息来辅助选择，统计目标点和邻域点的候选视差集合中目标点候选视差的数量，数量最多的视差为最终视差。具体算法流程见算法1

算法1

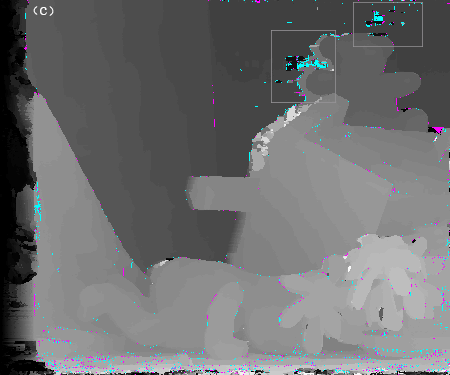
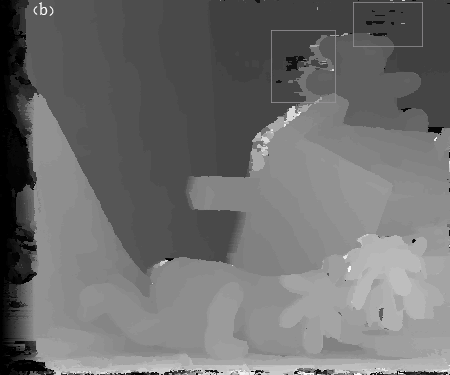
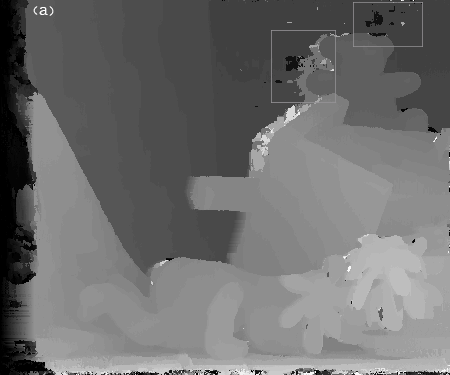
1. 记录匹配代价卷每个图像坐标下的最小代价值的N%范围内的前M个视差，记为候选代价卷V，V(u, v)表示坐标(u, v)的候选视差集，其中u表示行号，v表示列号，Vi(u, v)表示V(u, v)中的元素，Ci(u, v)表示Vi(u, v)的代价，N(u, v)表示V(u, v)中元素个数,D(u, v)表示(u, v)点最终确定的视差。
2. 若u==0或者N(u, v) ==1, 则D(u, v) = , 否则进行(3);
3. 设 若S < T1, 则将和存入集合H中，相同元素只存一次，用Hi表示H中的元素, Pi表示Hi的代价
4. 统计H中元素的个数n，若n==0，则, 否则，进行(5)
5. 若不存在 , 清空H，否则，进行（6）

统计H, V(u-1, v), V(u + 1, v)中Hi的数目Ni和对应的代价和Xi，, 若h唯一，则D(u, v) = h, 否则，清空H

设置N为105，M为4，图1（a），（b）分别是在本文框架下执行WTA策略和算法1获取

图1 不同视差计算策略下的视差图。（a）WTA的视差图；（b）算法1的视差图；（c）WTA和算法1的视差对比图；（d）原始图像

Fig.1 Disparity maps of different disparity calculating stratagies. (a)Disparity map of WTA; (b)Disparity map of algorithm 1;(c)Disparity comparision map of WTA and algorithm 1;(d)Original map

的视差图，（c）是（b）和（a）的视差对比图，其中蓝色区表示(a)图视差错误而（b）图视差正确的区域，红色区表示（a）图视差正确而（b）图视差错误的区域，其他区域表示（b）和（a）的正确性一致。可以发现，在重复纹理区（框选区域）算法1比WTA有更少的误匹配，表明了算法1的有效性。

2.2视差后处理

经过上述步骤后，在遮挡区和视差不连续区会存在大量的误匹配点，将它们检测出后需要利用邻域的视差进行插值填充。首先，利用左右一致性检测标记出左视差图的误匹配点，并进一步分为遮挡点和误匹配点：

(12)

式中，为左视差图，为右视差图，当像素p满足（12），则将其标记为合理点，否则，记其为误匹配点并做下一步划分，若d,满足公式（13）：

(13)

则将点记为误匹配点，否则记为遮挡点。

对于误匹配点，采取两轮不同的插值策略：第一轮，进行区域投票插值【CBCA】插值，区域投票利用自适应窗口中的频率最高的合理视差来进行插值，虽然可靠性高，但是由于位于遮挡区的点的自适应窗口较小并且含有的合理视差数目较少，导致区域投票策略可靠性下降，插值效率低，所以对于遮挡点，我们采取背景插值策略：

(14)

式中，，分别表示p点左侧和右侧第一个合理点，用限制搜寻的范围来控制可靠性。

对于误匹配点，采取区域投票插值策略。。。。。。

## 4.实验结果与分析

实验平台配置为Intel(R) Core(TM) i5-7300U CPU，2.60GHz，8GB内存。利用visual studio 2019IDEA，使用C++语言编程实现算法。数据集选用Middleburry基准测试平台的第二版的数据集上，包括Teddy、Cones、Venus、Tsukuba四对图片，并设置视差搜索范围分别为0~59、0~59、0~19、0~15px。

评价标准和Middleburry平台保持一致，即误匹配像素百分比，并且设置误差限为1个像素，当某个像素点的预测视差值和真实视差值的差距大于1时，该点被识别为误匹配像素，分别计算在非遮挡区、所有区域和视差不连续区的误匹配像素百分比，区域的选择通过掩膜图来实现，掩膜图为二值灰度图，分别为0和255，其中255表示需要计算的点，而0表示不计算的点。

4.1 结合邻域的视差计算策略的有效性

4.2联合插值策略的有效性

## 5.结论