## 基于新的代价计算方法和联合插值策略的立体匹配优化

## 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。

大量研究重点关注匹配代价计算和代价聚合方法的研究。匹配代价计算的经典方法包括：AD、BT、Census、ZNCC、梯度、互相关【】等。近些年在匹配代价计算上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。代价聚合主要考虑两个问题：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。

不论采用何种代价计算和聚合策略，产生的结果-视差仍存在各种不同的误差。主要存在于遮挡区和重复纹理区。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。

视差计算与视差优化的主要方法是插值法。。。。WTA，对于。。。问题能得到较好的解决，但是对于。。。。仍

基于WTA的优点方法：局部视差直方图的视差精化方法，实际上是统计局部出现次数最大的视差（祝世平，2015）；在视差选择阶段采用WTA-DP 相结合的动态规划进行视差选择，消除了传统动态规划固有的扫描线效应，并修改了动态搜索半径，降低了运算复杂度。（祝世平2016）。郭鑫（2019）等提出的可靠性差别视差修正算法，是先得出最小和次小匹配代价，并判定这两个代价是否大于给定阈值，若小于阈值证明视差值不可靠，对于不可靠的像素点，以该点为中心建立5x5矩形窗口，搜索窗口内的距离和颜色最接近的点，并把找到点的视差作为更新后的视差。这种方法的问题很明显，首先不同图像的阈值可能不同，其次在搜索窗内找到的点也不见得是可靠点。

本文关注立体匹配中视差计算和视差优化方法，提出了两个优化策略来进一步提高立体匹配的精度。首先，为了提升初始视差图的精度，提出了一种结合邻域的视差计算策略，对比WTA策略，使重复纹理区的误匹配明显降低；第二，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，我们对比了在立体匹配前两个步骤——代价计算和代价聚合分别采用多种算法时，本文视差计算与优化策略的性能，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.代价计算与代价聚合方法

由于本文针对局部立体匹配中的视差计算和视差优化方法进行改进，因此需要首先对广泛使用的代价计算和代价聚合方法进行介绍。并且，这些方法将在实验部分用于分析和验证本文所提出的后处理方法的有效性。

### 2.1 代价计算

这个步骤通过计算像素描述符的差异或者相似性来获取初始匹配代价卷。AD-Census将像素强度和局部区域像素的相对强度编码作为描述符，给定左图的一个像素点和视差，其代价计算公式为：

 （1）

其中，是像素点和其右图对应点的两个位串的汉明距离【Census】，是两点的RGB三通道的绝对差的均值：

 (2)

是控制离群值影响的参数，的公式如下：

 （3）

Census能够抵抗光照和噪声的影响并且在弱纹理区也有较好的表现，是因为它是通过对局部结构的比较来计算代价，但同样的原因，当遇到重复结构或相似纹理时，Census的代价计算会产生很大的歧义性；AD通过直接比较颜色差来获取代价，对纹理细节的感知更加敏锐，所以它适合作为重复纹理区的描述符，但是在无纹理和弱纹理区，AD表现不好。AD-Census融合了两者的优点，在重复纹理和弱纹理都有较好的表现。

梯度（Gradient）能有效刻画物体边缘，在代价计算中作为单个方法或者融合方法之一被研究【】【】。AD-Gradient对光照改变鲁棒并被广泛用于和立体匹配类似的光流估计中，并被【Guide】用作代价计算方法。本文采用的AD-Gradient的公式如下：

 (4)

式中，表示x方向的梯度值，为控制颜色项和梯度项权重的参数。

### 2.1 代价聚合

代价聚合可被看作对匹配代价卷的每个视差层的滤波操作，实质上是利用邻域信息来降低目标点视差的歧义性。自适应窗口法中的十字交叉法和自适应权重法中的引导滤波法作为各自所属类别中的经典算法，受到了广泛地研究。

十字交叉法根据颜色相似性和距离相近性为每个像素点构建自适应窗口。首先，计算目标点一个方向上（水平或者竖直）的支持臂长，假设选择水平方向，则计算左臂和右臂的长度，以左臂为例，公式如下：



其中，，L为预设的最大臂长，是目标像素点和臂上当前像素点的颜色差是否超过阈值的指示器，是臂上当前像素点和它前面一个像素点的差是否超过阈值的指示器，如下所示：





和分别是对颜色差和距离差施加限制，臂长最长不能超过，在内层距离范围内，选择的颜色差阈值大于外层距离范围的颜色差阈值，即对距离更远的点施加更严格的颜色限制，这一策略在增大支持窗口面积的同时限制引入不同视差层的点。对臂上的邻近像素点施加颜色限制。按同样的方式计算右臂长度。

然后，以水平臂上的每个点作为锚点，分别计算其上臂和下臂的长度，至此，目标点p的支持区域构建完毕，用分别表示左、右、上、下臂的长度，和分别表示水平臂和竖直臂上的点的集合，则可通过如下公式表示：



其中，



当然，也可以先计算目标点的竖直臂长，然后计算竖直臂上每个点的水平臂长，两者生成的支持区域可能不同，但没有性能上的优劣性。

引导滤波（待写）

深度学习网络MC-CNN-fast（我记得有论文是在深度学习之后再进行后处理，不知道是否可以提高其后处理的结果？）

## 3.本文算法

**3．1 结合邻域信息与候选视差的视差计算**

常用的WTA(Winner-Takes-All)视差计算策略，其基本思想是求取最小代价对应的视差,公式如下：

(11)

式中，为目标像素，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，但是易受噪声影响，尤其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。针对上述问题，我们利用邻域信息与候选视差来降低视差选择的歧义性。

本文首先通过统计分析验证“候选视差”这一思路的可行性。在双目匹配图像中截取视差错误较大的部分（弱纹理和重复纹理区）；然后将每个像素的匹配代价值从小到大排序，在最小匹配代价值\*105%的范围内选择最多4个匹配代价值，并取得其对应的视差（即，获得代价值接近最小代价的<=4个视差）；最后，统计这些候选视差与真实视差的正确匹配率。结果如表3-1所示。由表3-1可以看出，匹配代价最小的视差（候选0）正确率为58.4%，候选1-3的正确率总和为21.6%。根据统计值，如果能够利用候选视差来修正，理论上可以将视差的正确率提升到80%。但需要解决的问题是：如何在最小代价视差和候选视差中做出最优选择？

表3-1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | 视差的正确率 | | | |
|  | % | 候选0 | 候选1 | 候选2 | 候选3 |
| 只有一个视差 | 138/184=75% | 100% |  |  |  |
| 有两个视差 | 413/448=96% | 88.14% | 11.86% |  |  |
| 有三个视差 | 217/255=85% | 60.4% | 24.4% | 15.2% |  |
| 有四个视差 | 248/383=65% | 43.95% | 29.4% | 15.7% | 10.9% |
|  | 1016/1270=80% | 73% | 17.2% | 7.1% | 2.7% |
|  |  | 58.4% | 13.8% | 5.7% | 2.1% |

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。基本思想：考虑视差从左往右应该是连续变化的，对于当前像素有多个候选视差难以确定的情况，可以借助左、右临域像素的已确定视差或候选视差来辅助判别当前像素的最优视差。

首先，定义匹配代价卷每个图像坐标下的最小代价值的范围内的前M个视差为候选视差集，记作；候选视差集对应的代价值，为最小代价。令表示中元素个数，，表示像素点选定的视差。

接下来，考虑候选视差选择的三种情况，分别定义规则：

情况1：对于边缘像素u==0，以及只有一个候选视差N(u, v) ==1的情况，直接采用最小代价对应的视差

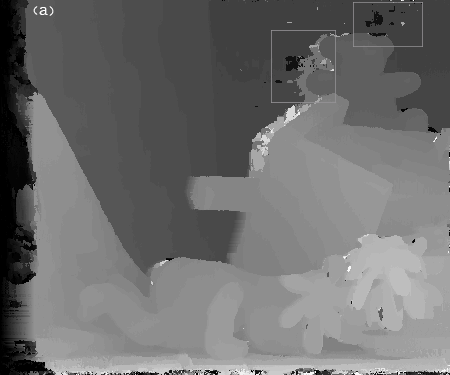
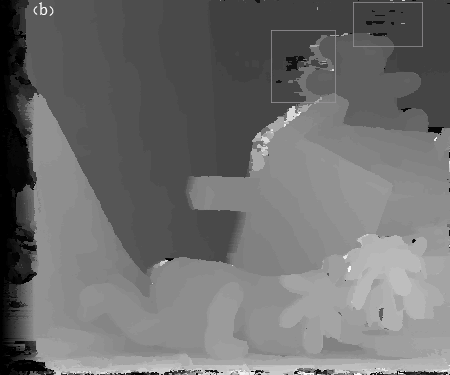
情况2：当候选集N(u, v) >1，对于候选视差集中任意检验两元素之间的差距，若任意两两距离差距过大说明该点的视差存在较大不确定性，需要借助邻域已经确定的视差来辅助该点的视差选择。此时，从候选视差中选择和已确定的左邻域视差最接近的视差

情况3：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，剔除后的当前点候选视差集为，相应的代价集为；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，此时目标点的候选集合为，统计中元素在集合中出现的次数，次数最多且代价最小的为最终视差

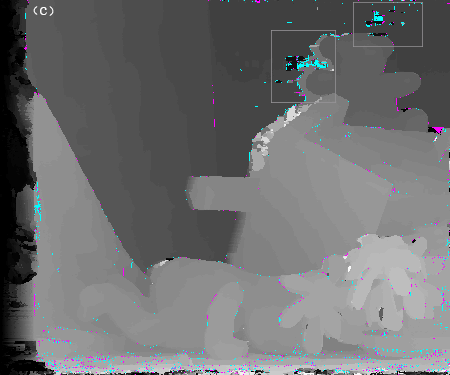
具体算法流程见算法1(待完成)

|  |  |
| --- | --- |
| 算法1：结合邻域信息与候选视差的视差计算 | |
| 输入： |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

设置a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（白色方框中）算法1比WTA有更少的误匹配，表明了算法1的有效性。

1. (b)

(c) (d)

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；蓝色区表示WTA视差错误而算法1图视差正确的区域，红色区表示WTA图视差正确而算法1视差图错误的区域；（d）原始图像

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

**3．2视差后处理**

经过视差计算，初始视差图在遮挡区存在大量的误匹配点需要处理。首先，利用左右一致性检测找出左视差图的错误点，判断公式如下：

 (12)

若满足上式，将其标记为合理点，否则，将其归为不合理点并做进一步划分，若满足下式：

 (13)

则将点记为误匹配点，否则记为遮挡点。

我们考虑了两种插值策略来组合插值—背景插值和区域投票插值。定义背景插值和区域投票插值得到的填充值分别为和，最终的填充值为。

背景插值是一种简单高效的插值策略，来自目标点左右两侧方向上的合理点，因为左右一致性检测主要针对遮挡区域，而遮挡区位于背景，所以背景插值选择靠近目标点且视差最小的视差点，公式如下：



式中，和分别是从目标点水平向左和向右范围找到的第一个合理点，用来控制搜索的距离，搜索距离越远，更容易找到合理点，但可靠性越低。

区域投票插值先构建目标点的十字交叉支持区域，然后用该区域内的合理视差值来构建直方图，横坐标为视差值，纵坐标为视差值的数目，最高的条柱所对应的视差记为，直方图中所有条柱的高度之和即该支持区域内的合理点的总数记为，在满足条件的情况下，将赋给：

，（5）

式中，和是两个阈值。区域投票策略在十字交叉区域内进行，保证了其较高的可靠性，但是在部分遮挡区，由于生成的十字交叉区域过小或者区域内合理点数目过少导致不能满足条件，造成插值失败，而简单的背景插值则可能完成次相同区域的插值。

本文提出的背景插值和区域投票相结合的组合插值策略，具体思想如下：当背景插值或区域投票没能找到填充值的情况下，将赋给或者；对于误匹配点，因为其大多位于非遮挡区，所以直接采用区域投票插值；而针对遮挡点，用背景插值和区域投票分别计算填充值，最终填充值选择公式如(X)所示：

经过联合插值，对于仍未完成插值的点（d=-1）,利用16方向插值【ADCensu】进行填充。最后为视差图执行的均值滤波。

## 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，包括Tsukuba、Venus、Teddy、Cones四对图片，并根据【CBCA】设置视差搜索范围分别为0~15、0~19、0~59、0~59px。

分别计算生成的视差图在非遮挡区、所有区域和视差不连续区的误匹配像素百分比，误差线设为1个像素。

**4.1 结合邻域的视差计算策略的有效性**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Tsukuba | | | Venus | | | Teddy | | | | Cones | | | |  | |
|  | nonocc | all | disc | nonocc | all | disc | nonocc | all | disc | nonocc | | all | disc | ave err | |
| 算法1 | 0.0151 | 0.0187 | 0.0801 | 0.0026 | 0.0071 | 0.0243 | 0.0565 | 0.1119 | 0.1410 | 0.0418 | | 0.1092 | 0.1043 | 0.0594 | |
| 算法2 | 0.0149 | **0.0186** | 0.0792 | **0.0023** | 0.0069 | 0.0237 | 0.0529 | **0.1082** | 0.1379 | 0.0412 | | 0.1072 | 0.1035 | 0.0580 | |
| 算法3 | **0.0140** | 0.0190 | 0.0745 | 0.0025 | **0.0048** | 0.0228 | **0.0518** | 0.1113 | 0.1346 | 0.0407 | | 0.1056 | 0.1025 | **0.0570** | |
| 算法4 | 0.0288 | 0.0328 | 0.1037 | 0.0078 | 0.0150 | 0.0989 | 0.0632 | 0.1175 | 0.1737 | **0.0336** | | **0.0917** | 0.0965 | 0.0719 | |
| 算法5 | 0.0297 | 0.0336 | 0.1018 | 0.0078 | 0.0151 | 0.0989 | 0.0638 | 0.1181 | 0.1761 | 0.0352 | | 0.0989 | 0.1008 | 0.0733 | |
| 算法6 | 0.0279 | 0.0329 | 0.0973 | 0.0075 | 0.0135 | 0.0927 | 0.0633 | 0.1244 | 0.1752 | 0.0340 | | 0.1001 | 0.0979 | 0.0722 | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 算法1：ADCensus+CBCA(4)+WTA+RV(4)+PI+MB | | | | | | | | | | | | | |
| 算法2：ADCensus+CBCA(4)+FromTopVm(6, 1.08)+RV(4)+PI+MB |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 算法3：ADCensus+CBCA(4)+FromTopVm(6, 1.08)+RV\_BG\_SmallPrior(4)+PI+MB |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 算法4：ADCensus+guideFilter+WTA+RV(4)+PI+MB |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 算法5：ADCensus+guideFilter+FromTopVm(6, 1.08)+RV(4)+PI+MB |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 算法6：ADCensus+guideFilter+WTA+RV\_BG\_SmallPrior(4)+PI+MB |  |  |  |  |  |  |  |  |  |  |  |  |  |

**CBCA为什么一定要用4次？运算代价太高**

**4.2联合插值策略的有效性**

## 5.结论

问题1：结合邻域的

我看到有论文里是对比不同误差精度下的性能



郭鑫 2019 三种测度融合的加权引导滤波匹配算法＊