doi:10.19306/j.cnki.2095-8110.2022.06.016

特征融合的双目立体匹配算法加速研究与实现

范亚博[1],王国祥[2],陈海军[2],冯　威[1]

(1. 西南交通大学地球科学与环境工程学院,成都 611756;
2. 中铁二院工程集团有限责任公司,成都 610031)

摘　要:随着图像分辨率和场景信息获取实时性需求的提高,业界对双目立体匹配算法的效率提出了更高的要求。针对该问题,提出了将 SAD 与 Census 变换特征融合的结果作为初始匹配代价,利用 SGM 算法进行代价聚合,采用赢家通吃策略计算视差,通过左右一致性检验检测出遮挡点并填充,使用中值滤波剔除异常值,最终获取优化后的视差图。采用统一计算设备架构(CUDA)对算法实现并行计算,针对立体匹配比较耗时的问题,该算法最大化地利用共享内存、寄存器内存以及 CUDA 流,实现了不同核函数之间的并行,大大提升了执行效率。结果表明,该算法在 Middlebury 立体匹配平台上,平均误匹配率下降了 8.05%;在 NVIDIA GeForce GTX 1650 平台上运行 450×375 分辨率的图像,比原始 SGM 算法快 687 倍,运行高分辨率图像时依然能够实现实时显示性能。

关键词:立体匹配;SAD;Census 变换;CUDA 加速;并行计算
中图分类号:TP391.4　　**文献标志码**:A　　**文章编号**:2095-8110(2022)06-0133-08

Research and Implementation Accelerating of Binocular Stereo Matching Algorithm Based on Feature Fusion

FAN Ya-bo[1], WANG Guo-xiang[2], CHEN Hai-jun[2], FENG Wei[1]

(1. Faculty of Geosciences and Environmental Engineering, Southwest Jiaotong University, Chengdu 611756, China;
2. China Railway Eryuan Engineering Group Co., Ltd., Chengdu 610031, China)

Abstract: With the increase in image resolution and the demand for real-time scene information acquisition, the industry has put forward higher demands on the efficiency of binocular stereo matching algorithms. To address this problem, the outcome of the fusion of SAD features and Census transform is proposed as the initial matching cost, cost aggregation is performed using SGM algorithm, parallax is calculated using a winner-take-all strategy, occlusion points are detected and filled by left-right consistency check, outliers are removed using median filtering, and finally the optimized parallax map is obtained. The CUDA is used to parallelize the algorithm. For the problem that stereo matching is relatively time-consuming, the algorithm maximizes the use of shared memory, register memory and the use of CUDA streams to achieve parallelism among different core functions, which greatly improves the execution efficiency. The results show that the algorithm reduces the average false match by 8.05% on the Middlebury stereo matching platform; it is 687 times faster than the original SGM algorithm when running 450×375 resolution images on the NVIDIA GeForce GTX 1650 platform, and it still achieves real-time display performance when

收稿日期:2021-11-12;**修订日期**:2022-01-17
基金项目:国家自然科学基金(42171429);四川省科技计划资助项目(2020GZYZF0010)
作者简介:范亚博(1993-),男,硕士研究生,主要从事视觉导航与图像处理方面的研究。
通信作者:冯威(1984-),男,博士,讲师,主要从事 GNSS 理论与应用、视觉导航方面的研究。

running high-resolution images.

Key words: Stereo matching; SAD; Census transform; CUDA acceleration; Parallel computing

0 引言

立体视觉是机器认识世界的重要手段。立体匹配是双目立体视觉技术中的一个重要环节,其通过左右两幅图像寻找匹配像素点,利用视差信息获取空间物体距离拍摄位置的距离[1]。双目立体匹配因其可靠、简便的特点,使其在机器人导航、自动驾驶、三维重建及距离测量等领域得到了广泛的应用。随着图像分辨率的提高以及实际应用场景中对数据信息实时性需求的愈发迫切,业界对立体匹配算法的实时性和准确性提出了更高的要求。

D. Scharstein 等[2]对经典的匹配算法进行了总结,立体匹配算法通常分为以下 4 步:匹配代价计算、代价聚合、视差计算和视差优化。目前,立体匹配算法根据算法特点可以分为局部、全局、半全局和基于深度学习的立体匹配算法[3]。全局算法通过最小化全局能量函数得到最优视差,但计算复杂度较高,难以满足实时性要求。局部算法精度劣于全局算法,但计算复杂度低,能满足实时的要求。半全局匹配(Semi-Global Matching,SGM)算法结合局部和全局匹配算法的优点,具有复杂度低、效率高和易于实现等优点[4]。近来,卷积神经网络(Convolutional Neural Networks,CNN)[5]被应用于立体匹配算法中,实现了更高的匹配精度,但 CNN 巨大的计算消耗和密集度,使其很难应用于硬件资源有限的实时立体匹配系统[6]。

Census 变换算法凭借其易于并行的优点成为主流的代价计算方法,但传统的 Census 变换过于依赖中心像素点,且受支持窗口尺寸的影响[7]。为提高 Census 变换算法的性能,专家提出了很多改进策略,主要有:1)采用变换窗口内所有像素的加权平均值代替中心像素灰度值[8];2)采用中心对称 Census 变换以及稀疏 Census 变换[9];3)改变 Census 变换窗口的尺寸和形状;4)融合其他具有互补特性的代价计算方法[10-11]。这些算法虽然能达到很高的精度,但由于算法的计算复杂度较高,仅依靠中央处理器(Central Processing Unit,CPU)计算难以满足实时性需求。近来,基于图形处理器(Graphics Processing Unit,GPU)并行计算平台对 SGM 算法进行并行加速取得了不错的效果[1,12]。因此,利用 GPU 加速已成为实时获取匹配信息的可行途径。基于此,本文提出了利用统一计算设备架构(Compute Unified Device Architecture,CUDA)对双目立体匹配算法进行并行加速,从而满足实时性要求。

1 双目立体匹配算法描述

算法具体实施步骤如下:首先,对校正后的图像对进行直方图均衡化预处理,在此基础上,采用灰度差绝对值之和(Sum of Absolute Differences,SAD)融合传统的 Census 变换以替代单一的匹配代价计算算法。利用四路径 SGM 算法进行代价聚合,使用赢家通吃(Winner-Takes-All,WTA)优化策略进行视差计算,最终通过左右一致性检验对遮挡点进行填充以及利用中值滤波剔除异常值,继而得到优化后的视差图。基于 CUDA 加速的双目立体匹配算法处理流程如图 1 所示。

1.1 特征融合初始代价计算

SAD 算法具体流程为:用 i、j 遍历窗口 W 中每个像素,每进行一轮计算后 d 的数值增加 1,以使右图像中的窗口在视差范围内移动。移动过程中,两个子窗口计算的像素灰度差绝对值之和最小时,即找到了与左图像窗口相匹配的像素块,此时的 d 为视差值。SAD 算法计算公式为

$$C_{\mathrm{SAD}}=\sum_{i,j\in W}\left|I_{\mathrm{L}}(x+i,y+j)-I_{\mathrm{R}}(x+i+d,y+j)\right| \tag{1}$$

式中,$I_{\mathrm{L}}(x,y)$、$I_{\mathrm{R}}(x,y)$ 分别为左、右图像中 (x,y) 位置的像素灰度值;d 为视差范围。

Census 变换[13]是一种局部非参数变换算法,以某一像素点为中心取一个 $m\times n$(一般 m、n 为奇数)的变换窗口,通过将邻域窗口内的像素灰度值与窗口中心的像素灰度值进行比较,将得到的布尔值连接成一个二进制比特串。对左、右图像得到的两个比特串进行异或运算,统计计算结果为 1 的个数,即为汉明距离(Hamming distance),最后用汉明距离代替原来中心像素的灰度值。基于 Census 变换的初始匹配代价计算过程如图 2 所示。

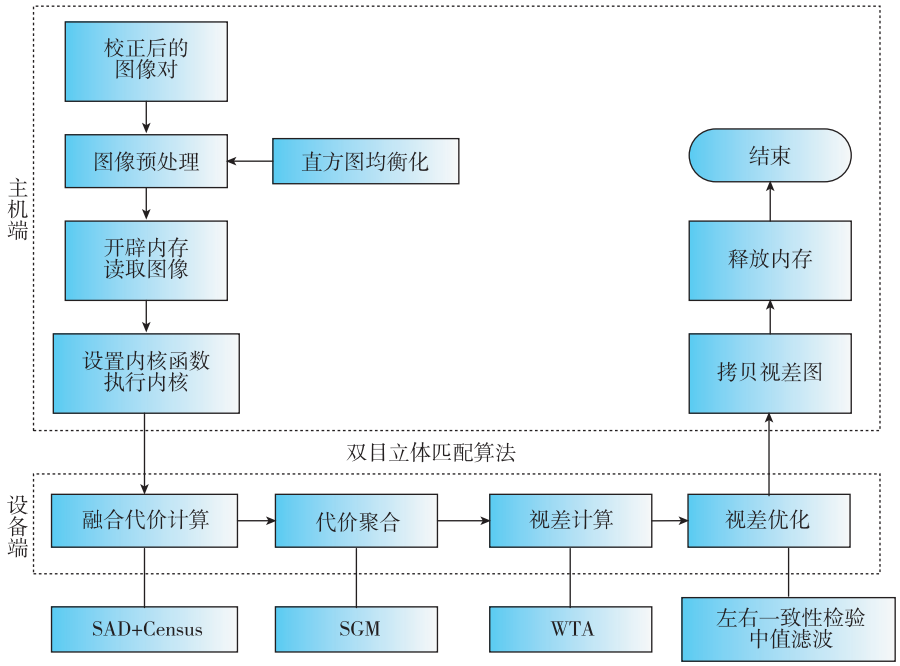


图 1　本文算法流程图

Fig. 1　Flow diagram of SAD-Census algorithm

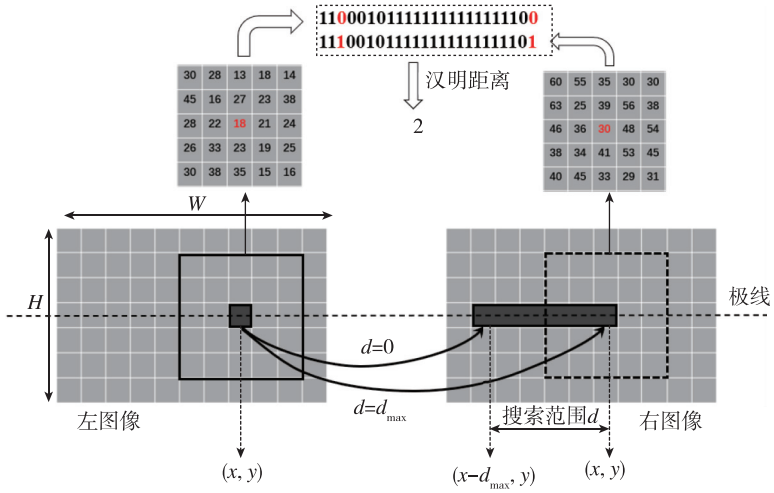


图 2　基于 Census 变换的初始代价计算示意图

Fig. 2　Calculation of the initial costs based on Census transform

SAD 算法匹配速度快，但对曝光度和光照变化较为敏感。Census 变换是基于窗口内的相对亮度差，对整体的明暗变化并不敏感，在重复纹理区域易产生误匹配。因此，将两个算法进行融合不但可以提高匹配精度，而且对光照条件和曝光差异也可以保持良好的鲁棒性。同时，SAD 和 Census 变换都是基于局部窗口运算，每个像素之间互不影响，可以独立运算，该特性使其可以设计多线程并行计算模型，但融合时需要考虑两个算法在不同场景下的权重。为了使得到的初始代价值更可靠，将 SAD 和 Census 变换窗口的尺寸考虑进公式中，特征融合后的初始代价值计算公式为

$$C(p,d)=2-\exp\left(-\frac{C_{\mathrm{SAD}}}{50\times a^{2}}\right)-\exp\left(-\frac{C_{\mathrm{Census}}}{0.6\times m\times n}\right) \tag{2}$$

式中，C 是代价值；a 是 SAD 算法窗口的大小；m、n 表示 Census 变换窗口的大小。

1.2 匹配代价聚合

由于初始代价计算只考虑了局部窗口信息，易受噪声的影响。为了获得更准确的代价值，本文采用 SGM 算法[4]进行代价聚合。算法主要是将二维图像的优化问题转换为多条路径的一维最优问题，使用从整个图像各个方向聚合的路径优化来近似全局能量。聚合路径的数量会影响视差图的质量和算法的性能。对于像素点 p，沿着某条路径 r 在视差为 d 时的代价聚合公式为

$$L_r(p,d)=C(p,d)+\min\begin{Bmatrix}L_r(p-r,d)\\L_r(p-r,d\pm1)+P_1\\\min\limits_j L_r(p-r,j)+P_2\end{Bmatrix}-\min\limits_j L_r(p-r,j) \tag{3}$$

式中，第一项为初始匹配代价值；第二项为路径 r 上前一个像素点的最小匹配代价，P_1 和 P_2 为视差不连续惩罚因子，且 $P_1>P_2$；第三项是为了防止聚合代价值过大而减去上一像素的最小代价值。

将各个路径上的代价值相加即为该像素点的最终代价值，计算公式为

$$E(p,d)=\sum_r L_r(p,d) \tag{4}$$

1.3 视差计算与视差优化

对代价聚合步骤得到的匹配代价，采用比较经典的 WTA 算法进行视差计算，即为每个像素选择所有视差下代价值中最小的代价值所对应的视差作为最优视差。

为了对视差计算步骤得到的视差图做进一步的优化处理，获得更加准确和密集的视差图，本文采用左右一致性检验有效地检测遮挡点和误匹配点，遮挡区域大多来自于背景，所以往往用背景视差值填充。由于图像对本身存在噪声以及填充后的视差可能存在异常值，采用中值滤波进行剔除，最终得到优化后的视差图。

2 双目立体匹配算法并行加速

利用 CUDA 进行算法加速设计时，首先要最大程度地利用 GPU 多线程的优势；其次合理使用各种设备内存，优化内存访问速度；最后利用多个 CUDA 流同时启动多个内核，实现核函数之间的并行加速计算。CUDA 程序的并行层次主要分核函数内部的并行和核函数外部的并行。

2.1 核函数内部的并行

核函数内部并行的高性能很大程度上取决于对各种设备内存的合理使用。由于设备内存的带宽可能是性能瓶颈，高效的 CUDA 代码应该促进共享内存和寄存器内存上的数据重复利用。由于图像对数据从 CPU 传入 GPU 时需要占据较大的显存空间，一般是存放在全局内存中。非连续的、重复的访问全局内存，会使算法的性能大大下降。

算法优化的思路如下：设图像对的宽、高分别为 W 和 H，匹配窗口的大小为 $m\times n$，视差范围为 d。计算 SAD 特征和 Census 变换时，每个线程对应一个像素点，为每个线程块(Block)分配 32×32 个线程，考虑到线程块的边缘线程无法通过共享内存访问相邻 Block 的数据，因此相邻 Block 之间需要一定的重叠，整张图像被划分为 $(W-[m/2])/(32-[m/2])\times(H-[n/2])/(32-[n/2])$ 个 Block([*]表示向下取整运算)。左、右图像每个线程块中需要读入共享内存的数据如图 3 蓝色矩形所示，红色矩形表示当前操作要处理的线程块，右图中的红色矩形在视差范围内向右移动。利用同步函数进行处理后，线程块内的所有线程都可以从共享内存中读取数据进行计算。算法将大量重复、非连续的访问全局内存改为重复访问速度更快的共享内存，大大地提高了算法的执行效率。

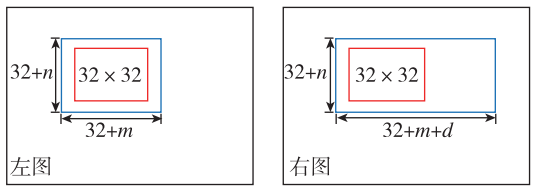


图 3 SAD 和 Census 变换算法加速模型

Fig. 3 SAD and Census transform algorithm acceleration model

2.2 核函数外部的并行

要想使算法获得更高的性能，需尽量减少主机与设备之间的数据传输以及在主机中进行的计算。因此，算法设计时把立体匹配算法比较耗时的 4 个步骤都放在设备中进行计算。该方式可以减少主机与设备之间数据传输带来的额外开销。同时，利用多个非默认 CUDA 流(stream)实现核函数之间的并行，进一步提高了设备利用率。

SAD 算法和 Census 变换都是基于输入的图像对

进行操作，两者相互独立，可以并行计算，但完成不同的 CUDA 流彼此间可能需要等待，因此需要同步操作。在核函数的执行配置中必须包含一个流对象，核函数的调用方式为：kernel <<<grid，block，N_shared，stream_id>>>(函数参数)。实验中发现，核函数之间的并行能够缩短算法总运行时间。串行运行和多流并发执行性能对比，如图 4 所示。

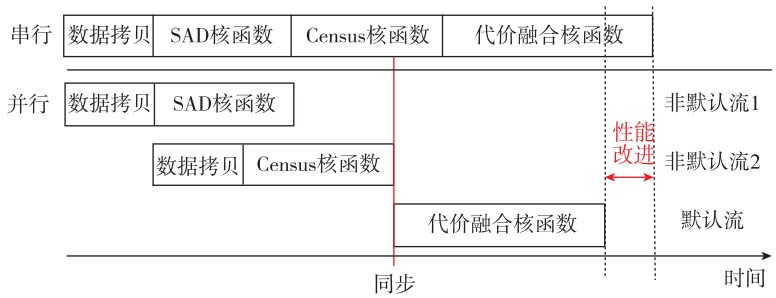


图 4　串行运行和多流并发执行性能对比

Fig. 4　Performance comparison of serial operation and multi-stream concurrent execution

考虑到每个像素的聚合代价值都是上、下、左、右 4 条路径下的代价值累加的结果，在一条指定路径下，需按顺序依次执行每个像素的代价聚合，这在一定程度上影响了算法的并行性。但每条路径只是聚合方向不同，其他操作完全相同。因此，几条路径间是彼此独立执行的，所以，为每条路径单独设置一个核函数，利用共享内存且执行同步操作，待所有核函数执行完毕后，把各个路径的代价值相加，得到最终的代价聚合值。

3　实验结果与分析

3.1　测试平台和测试数据集

为了充分测试算法的匹配精度和执行效率，对实验环境进行搭建，实验环境和平台信息如表 1 所示。采用 Middlebury 立体匹配算法测试平台[14]提供的 Venus、Teddy 及 Cones 标准彩色图像对算法进行实验，以此来评价算法的准确性和实时性。

表 1　实验环境平台介绍

Tab. 1　Introduction of the experimental environment platform

设备名称	Lenovo Legion Y7000
CPU 处理器	Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz　2.59GHz
显卡名称	NVIDIA GTX 1650
操作系统	Linux Ubuntu 18.04
编程环境	Visual Studio 2017
编程语言	C++

3.2　立体匹配算法性能分析

本文采用速度评估标准和精度评估标准测试 CUDA 算法的优化效率和匹配准确性。速度评估标准指算法的运行时间，精度评估标准是与 Middlebury 数据集提供的标准视差图进行比较，计算出整张图像全区域的误匹配率。

在测试算法的效率优化性能时，对实验过程中的变量进行控制，将各图像对的视差范围依次设置为 64 像素和 128 像素。经过多次实验，匹配效果最佳时，其他参数设置如下：SAD 窗口尺寸为 5×5，Census 变换窗口尺寸为 9×7，聚合代价路径数为 4。利用本文算法对不同分辨率的图像对进行测试，结果如表 2 所示。

表 2　不同视差情况算法匹配耗时

Tab. 2　Algorithm matching time for different parallax situations

图像名称	分辨率	视差 $d=64$	视差 $d=128$
Tuskuba	384×288	1.5ms(666.7fps)	2.4ms(416.7fps)
Venus	434×383	2.4ms(416.7fps)	3.7ms(270.3fps)
Cones	450×375	2.6ms(384.6fps)	3.8ms(263.2fps)
Ours	640×480	3.6ms(277.8fps)	6.0ms(166.7fps)
KITTI	1241×376	5.4ms(185.2fps)	8.9ms(112.4fps)

从表 2 可以看出，本文所提立体匹配算法对于不同分辨率的图像基本上都能实现实时匹配。当图像分辨率为 384×288，视差范围为 64 像素时，帧率最高可达到 666.7fps；当图像分辨率为 1241×376，视差范围为 128 像素时，帧率较低，可达到 112.4fps。表 2 中的结果表明，基于 CUDA 加速的立体匹配算法具有较好性能，在处理较大图像分辨率图像时也可满足实时性要求。

为了更全面地验证本文算法的效率，将该算法

与其他加速算法进行比较。虽然不同算法的运行环境均不相同，但平台的性能差异相差不大。实际上算法自身的并行性、算法的优化策略以及算法中涉及的一些参数都会影响运行环境平台的性能。在图像分辨率和视差范围相同的情况下比较算法的测试结果，如表 3 所示。

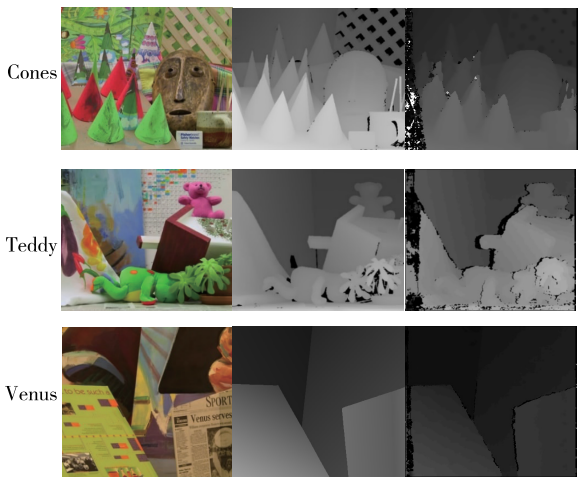
表 3 本文算法与其他加速算法的对比

Tab. 3 Comparison of this algorithm with other acceleration algorithms

算法描述	图像分辨率	视差/像素	运行环境	帧率/fps
本文算法	450×375	64	NVIDIA GeForce GTX 1650 GPU	384.6
	450×375	128		263.2
	640×480	64		277.8
	1241×376	128		112.4
分层匹配 SGM 算法[12]	450×375	64	NVIDIA GeForce GTX 1070 GPU	322.6
	640×480	64		212.8
稀疏 Census 变换[15]	450×375	50	NVIDIA GTX 280 GPU	75
原始 SGM 算法[4]	450×375	64	Opteron@2.2 GHz CPU	0.56

由表 3 可知：本文所提算法的实时性优于其他算法，在 NVIDIA GeForce GTX 1650 平台上实现了非常高的实时性能。处理 450×375 图像分辨率，视差范围为 64 时，最高可达 384.6fps 的匹配效率，比原始的 SGM 算法快 687 倍。与稀疏 Census 变换算法相比，可实现 5.1 倍的加速比。除去平台自身性能优势，加速效果也很不错。

为了验证算法的匹配精度，图 5 所示为算法的测试结果。由于 Venus 图像的纹理丰富，生成的视差图边缘较准确；而对于弱纹理较多的 Teddy 图像，部分叶子连成一体，但能较大程度上恢复出玩具熊和房子的轮廓；Cones 视差图中圆锥和面具模型的轮廓都比较清晰，边缘部分略有噪点，这进一步说明融合算法的准确性得到提高。



(a)左图像 (b)标准视差图 (c)视差图

图 5 Middlebury 测试数据集实验结果

Fig. 5 Experimental results of the Middlebury benchmark images

为了更好地评估生成视差图的效果，图 6 所示为各图像在不同算法下的全区域误匹配率。通过计算所有区域下各个测试集图像的误匹配率的平均值，得出本文算法的平均误匹配率为 10.4%。由图 6 也可知，本文算法的立体匹配精度高于 Edge-Gray 算法[16]和 V-SAD 算法[17]，平均误匹配率分别下降了 8.05% 和 3.22%，并且 Edge-Gray 算法和 V-SAD 算法在执行高分辨率图像时，实时性效果远不如本文所提算法。

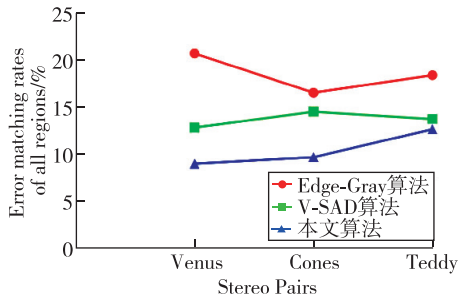


图 6 不同算法的视差图误匹配率

Fig. 6 Disparity map mismatching rates of different algorithms

此外，为了进一步验证算法在复杂道路场景下的准确性和实时性，本文测试了 KITTI 2012 数据集中自动驾驶场景下的图像，得到的视差图如图 7 所示(未对孔洞插值补全)。面对不同的光照和遮挡环境，本文算法得到的视差图可以清楚地看到路上行人、街道两旁树木以及汽车的轮廓。由上文可知，对于 1241×376 分辨率的室外道路图像，在 128 视差范围下能达到 112.4fps 的实时性能。由此可以得出该算法在复杂的室外道路场景下仍具有较

好的适用性。

(a)左图

(b)视差图

图 7　室外道路场景图实验结果

Fig. 7　Experimental results of outdoor road scene graphs

4　结论

针对传统双目立体匹配算法计算复杂度高且难以满足实时性要求的问题，本文提出了将 SAD 和 Census 变换特征融合的结果作为初始匹配代价，并对立体匹配算法耗时的 4 个步骤设计了基于 CUDA 的并行计算加速模型。算法分析与实验结果表明：

1)利用 SAD 和 Census 变换特征融合计算代价可以改善 Census 变换过于依赖中心像素点带来的影响，并且它们都是基于局部窗口运算，每个像素之间独立运行，使其可以设计并行计算模型。

2)数据传输过程中，最大化地利用共享内存、寄存器内存以减少全局内存的读取次数，使用多个 CUDA 流操作实现了不同核函数之间的并行。将本文算法与相关加速算法进行对比实验，结果表明，所提算法平均误匹配率下降了 8.05%，且比原始 SGM 算法快 687 倍。

3)算法在运行高分辨率图像(1241×376)时依然能够达到很好的实时显示性能(112.4fps)。同时，面对复杂的室外道路场景时，本文算法的准确性和实时性也表现突出。

参考文献

[1] Hernandez-Juarez D, Chacón A, Espinosa A, et al. Embedded real-time stereo estimation via Semi-Global Matching on the GPU [J]. Procedia Computer Science, 2016, 80(C): 143-153.

[2] Scharstein D, Szeliski R, Zabih R. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms[C]// Proceedings of IEEE Workshop on Stereo and Multi-Baseline Vision (SMBV 2001), 2002: 7-42.

[3] 赵晨园，李文新，张庆熙. 双目视觉的立体匹配算法研究进展[J]. 计算机科学与探索，2020，14(7)：1104-1113.
Zhao Chenyuan, Li Wenxin, Zhang Qingxi. Research and development of binocular stereo matching algorithm [J]. Journal of Frontiers of Computer Science and Technology, 2020, 14(7): 1104-1113(in Chinese).

[4] Hirschmüller H. Accurate and efficient stereo processing by semi-global matching and mutual information[C]// Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2005: 20-26.

[5] Žbontar J, Lecun Y. Stereo matching by training a convolutional neural network to compare image patches[J]. The Journal of Machine Learning Research, 2016, 17(1): 2287-2318.

[6] Lu Z, Wang J, Li Z, et al. A resource-efficient pipelined architecture for real-time semi-global stereo matching[J]. IEEE Transactions on Circuits and Systems for Video Technology, 2021(99): 1.

[7] 祝世平，闫利那，李政. 基于改进 Census 变换和动态规划的立体匹配算法[J]. 光学学报，2016，36(4)：216-224.
Zhu Shiping, Yan Lina, Li Zheng. Stereo matching algorithm based on improved census transform and dynamic programming[J]. Acta Optica Sinica, 2016, 36(4): 216-224(in Chinese).

[8] Spangenberg R, Langner T, Rojas R. Weighted semi-global matching and center-symmetric census transform for robust driver assistance[C]// Proceedings of International Conference on Computer Analysis of Images and Patterns. Springer, Berlin, Heidelberg, 2013: 34-41.

[9] Humenberger M, Zinner C, Kubinger W. Performance evaluation of a census-based stereo matching algorithm on embedded and multi-core hardware[C]// Proceedings of 6th International Symposium on Image and Signal Processing and Analysis, 2009: 388-393.

[10] Mei X, Sun X, Zhou M, et al. On building an accurate stereo matching system on graphics hardware[C]// Proceedings of 2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops), 2011: 467-474.

[11] 程德强，李海翔，寇旗旗，等. 融合边缘保持与改进代价聚合的立体匹配算法[J]. 中国图象图形学报，2021，26(2)：438-451.
Cheng Deqiang, Li Haixiang, Kou Qiqi, et al. Stereo

matching algorithm based on edge preservation and improved cost aggregation[J]. Journal of Image and Graphics, 2021, 26(2): 438-451(in Chinese).

[12] 李迎松. 摄影测量影像快速立体匹配关键技术研究[D]. 武汉: 武汉大学, 2018.
Li Yingsong. Research on key techniques of fast stereo matching for photogrammetric images[D]. Wuhan: Wuhan University, 2018(in Chinese).

[13] Zabih R, Woodfill J. Non-parametric local transforms for computing visual correspondence[C]// Proceedings of European Conference on Computer Vision (ECCV), 1994: 151-158.

[14] Scharstein D, Szeliski R. High-accuracy stereo depth maps using structured light[C]// Proceedings of 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003: 195-202.

[15] Humenberger M, Zinner C, Weber M, et al. A fast stereo matching algorithm suitable for embedded real-time systems[J]. Computer Vision and Image Understanding, 2010, 114(11): 1180-1202.

[16] 张一飞, 李新福, 田学东. 融合边缘特征的 SAD 立体匹配算法[J]. 计算机工程, 2020, 46(4): 236-240, 246.
Zhang Yifei, Li Xinfu, Tian Xuedong. SAD stereo matching algorithm combining edge features[J]. Computer Engineering, 2020, 46(4): 236-240, 246(in Chinese).

[17] 李一能, 曾庆化, 张月圆, 等. 融合明度特征的 V-SAD 立体匹配算法[J]. 导航定位与授时, 2021, 8(4): 90-97.
Li Yineng, Zeng Qinghua, Zhang Yueyuan, et al. V-SAD stereo matching algorithm combining value features[J]. Navigation Positioning and Timing, 2021, 8(4): 90-97(in Chinese).

(编辑:孟彬)