E-mail: jig@radi.ac.cn
Website: www.cjig.cn
Tel: 010-64807995

中国图象图形学报
JOURNAL OF IMAGE AND GRAPHICS

中图法分类号: TP301.6　　**文献标识码:** A　　**文章编号:** 1006-8961(2018)06-0874-00

论文引用格式: Lyu N Q, Song G H, Yang B W. Semi-global stereo matching algorithm based on feature fusion and its CUDA implementation [J]. Journal of Image and Graphics, 2018, 23(6): 0874-0886. [吕倪祺, 宋广华, 杨波威. 特征融合的双目半全局匹配算法及其并行加速实现[J]. 中国图象图形学报, 2018, 23(6): 0874-0886.] [DOI: 10.11834/jig.170157]

特征融合的双目半全局匹配算法及其并行加速实现

吕倪祺, 宋广华, 杨波威

浙江大学航空航天学院空天信息技术研究所, 杭州　310027

摘　要: **目的**　在微小飞行器系统中, 如何实时获取场景信息是实现自主避障及导航的关键问题。本文提出了一种融合中心平均 Census 特征与绝对误差(AD)特征、基于纹理优化的半全局立体匹配算法(ADCC-TSGM), 并利用统一计算设备架构(CUDA)进行并行加速。**方法**　使用沿极线方向的一维差分计算纹理信息, 使用中心平均 Census 特征及 AD 特征进行代价计算, 通过纹理优化的 SGM 算法聚合代价并获得初始视差图; 然后, 通过左右一致性检验检查剔除粗略视差图中的不稳定点和遮挡点, 使用线性插值和中值滤波对视差图中的空洞进行填充; 最后, 利用 GPU 特性, 对立体匹配中的代价计算、半全局匹配(SGM)计算、视差计算等步骤使用共享内存、单指令多数据流(SIMD)及混合流水线进行优化以提高运行速度。**结果**　在 Quarter Video Graphics Array(QVGA)分辨率的 middlebury 双目图像测试集中, 本文提出的 ADCC-TSGM 算法总坏点率较 Semi-Global Block Matching(SGBM)算法降低 36.1%, 较 SGM 算法降低 28.3%; 平均错误率较 SGBM 算法降低 44.5%, 较 SGM 算法降低 49.9%。GPU 加速实验基于 NVIDIA Jetson TK1 嵌入式计算平台, 在双目匹配性能不变的情况下, 通过使用 CUDA 并行加速, 可获得 117 倍以上加速比, 即使相较于已进行 SIMD 及多核并行优化的 SGBM, 运行时间也减少了 85%。在 QVGA 分辨率下, GPU 加速后的运行帧率可达 31.8 帧/s。**结论**　本文算法及其 CUDA 加速可为嵌入式平台提供一种实时获取高质量深度信息的有效途径, 可作为微小飞行器、小型机器人等设备进行环境感知、视觉定位、地图构建的基础步骤。

关键词: 双目视觉; census 特征; 半全局匹配; CUDA 加速; 并行计算

Semi-global stereo matching algorithm based on feature fusion and its CUDA implementation

Lyu Niqi, Song Guanghua, Yang Bowei

Institute of Aerospace Information Technology, School of Aeronautics and Astronautics, Zhejiang University, Hangzhou　310027, China

Abstract: **Objective** In unmanned aerial vehicle systems, estimation of scene information in real time is a key issue in conducting automatic obstacle avoidance and navigation. A binocular stereo vision system is an effective means to obtain scene information; this system simulates the working principle of the human eyes by using two cameras to capture the same sense at the same time and generates a disparity map by using a stereo matching algorithm. In this work, we propose ADCC-TSGM, a novel texture-optimized semi-global stereo matching algorithm based on the fusion of absolute difference (AD) fea-

收稿日期: 2017-11-10; **修回日期:** 2017-12-26; **预印本日期:** 2018-01-04

基金项目: 国家自然科学基金项目(61272467, 61501399); 装备预研教育部联合基金(重点)项目

第一作者简介: 吕倪祺(1994—), 男, 浙江大学航空航天学院空天信息技术专业硕士研究生, 主要研究方向为计算机视觉。E-mail: lvniqi@gmail.com

通信作者: 宋广华, 教授, E-mail: ghsong@zju.edu.cn

Supported by: National Natural Science Foundation of China (61272467, 61501399)

ture and center average census feature. Efforts are made to speed up the algorithm through CUDA parallel acceleration. **Method** First, a one-dimensional difference method is used to calculate the texture information along the epipolar line, the center average census feature and AD feature are exploited to conduct the cost computation, and the global stereo matching algorithm is texture-optimized to aggregate the cost and obtain the initial disparity. Second, left-right consistency check is used to detect unstable pixels and occlusion pixels, and linear interpolation and median filter method are used to fill the holes of the disparity map. Lastly, to improve the running speed, we optimize the code of GPU acceleration for each step of the stereo matching. The time consumption of memory access is considered in the feature calculation of types, such that center average census is much higher than that of computation, and a large number of data-intensive computing tasks are conducted between adjacent threads. Consequently, we divide the dataset of the entire thread block into four regions, copy them into a shared memory, and use the shared memory for computation to reduce the overhead of memory accessing. A single thread can simultaneously handle two consecutive disparity calculations by using SIMD instructions. When the GPU is processing, the CPU is basically idle. Therefore, a hybrid pipeline is designed to fully utilize the computing resources of the embedded platform. **Result** To demonstrate the effectiveness of the proposed algorithm, we use NVIDIA Jetson TK1 developer kit, which has a quad-core ARM Cortex-A15 CPU, a Kepler GPU with 192 CUDA cores, and 2 GB memory, as the embedded computing platform to conduct experiments on Middlebury stereo datasets that have been resized to QVGA resolution. With the actual application scenarios and resolution of images, the maximum disparity for each algorithm is set to 64, and the block matching window size of SGBM and BM is set to 9×9. The texture penalty coefficients $\varepsilon 1$ and $\varepsilon 2$ in the proposed algorithm are set to 0.25 and 0.125, respectively. Experimental results show that the total bad-pixel rate and the average error rate of the proposed algorithm are significantly lower than those of BM, SGBM, and SGM, respectively. The total bad-pixel rate of the ADCC-TSGM algorithm is 73.9% lower than that of BM algorithm, 36.1% lower than that of SGBM algorithm, and 28.3% lower than that of SGM algorithm. The average error rate of the proposed algorithm is 83.2% lower than that of the BM algorithm, 44.5% lower than that of the SGBM algorithm, and 49.9% lower than that of the SGM algorithm. In particular, the use of center average census in feature matching can reduce the bad-pixel and error rates. The texture-based optimization can adaptively increase the penalty coefficient in low-texture regions and reduce the average error rate from 6.62 to 4.84. The post-processing method, including disparity consistency check and hole filling, can reduce the total bad-pixel rate from 14.46 to 7.12. Through GPU parallel acceleration, the CUDA implementation of the proposed algorithm becomes hundreds of times faster than that of pure CPU implementation without any loss in the quality of disparity map. Compared with SGBM, which has been optimized by using SIMD and multi-core parallel method, our proposed algorithm has a running time that is reduced by 85%. For QVGA resolution, the frame processing rate is as high as 31.8 FPS. **Conclusion** The proposed algorithm outperforms existing algorithms, such as BM, SGM, and SGBM, which have been used in industries. The CUDA-accelerated implementation of the proposed algorithm provides an effective and feasible method to obtain high-quality disparity information and can be used as a basic means of environmental perception, visual positioning, and map construction for real-time embedded applications, such as micro-aircraft systems.

Key words: stereo vision; census feature; semi-global matching; CUDA acceleration; parallel computing

0　引　言

双目立体匹配算法可分为特征匹配和稠密匹配。特征匹配使用特征点进行左右视图的像素点匹配,计算精度较高,但所得的深度信息稀疏,并不适合复杂场景。对于稠密匹配算法,Scharstein D 进行了详细分类和评价[1],提出了现有立体匹配算法的四大模块,即匹配代价计算、代价/支持聚合、视差计算/最优化以及视差校正。一般而言,稠密匹配的双目视觉处理流程图如图 1 所示。

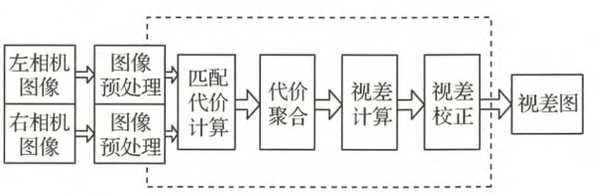


图 1　双目视觉处理流程图

Fig. 1　The flow diagram of the binocular stereo vision

对于匹配代价计算,大部分使用的是灰度相关

性算法，如绝对误差和(SAD)、误差平方和(SSD)。Kim 等人提出了分层互信息[2]，用于视觉匹配，减小了光照对于匹配的影响。Muquit 等人[3]创新地使用了频域相位信息与空间域平移以计算对应点处的视差值；在此基础上，提出了对任意形状的表面进行重构的算法，并采用多尺度搜索策略对孤立点进行检测和校正，得到可靠的亚像素级精度。

对于代价聚合，局部算法一般采用固定窗口进行计算；而全局算法，如图割算法[4]则通过考虑全图范围内的匹配代价直接计算获得视差信息。Zhang 等人[5]提出了快速自适应区域的双目匹配算法，为不同图像选择合适大小和形状的支持窗体。Mei 等人[6]使用了自适应窗口，并结合了多个代价函数，具有很高的实用化价值。

卷积神经网络(CNN)技术被认为是一种提取图像局部特征的有效手段。Zbontar 与 LeCun[7]通过使用一个 5 层卷积网络提取左右视图的特征，使用半全局匹配(SGM)进行代价匹配，取得了当时的最佳性能。在此基础上，Li[8]使用 3D 下的最小生成树(MST)进行聚合区域计算，取得了一定提升。然而这些高性能算法需要极大的计算量，难以在实际的嵌入式设备上部署。

应用方面，Hrabar[9]等人研究了使用双目视觉和光流相结合的方式，进行导航定位。Stefanik[10]使用双目视觉系统进行快速地形测绘，大大降低了系统的重量，提高了部署速度，取得了良好的结果。DJI 利用基于 FPGA 及 BM 算法实现的深度传感器[11]能产生 20 Hz 的深度图，然而分辨率只有 QVGA 级别，且质量一般，难以满足 3 维重建需求。

1 本文算法

本文使用中心平均 census 特征与 AD 特征作为匹配特征，使用基于纹理优化的 SGM 作为代价聚合算法，处理流程如图 2 所示。

1.1 图像预处理

首先，从一组平行的双目相机中采集左右图像对；然后，通过旋转矩阵和平移矩阵分别对图像进行重映射，再进行平滑滤波，以获得满足极线约束的图像对。图像预处理所需的镜头内外参数以及旋转与平移矩阵可由张正友标定法[12]并借由 OpenCV 标定函数求得，标定矫正后结果如图 3 所示。

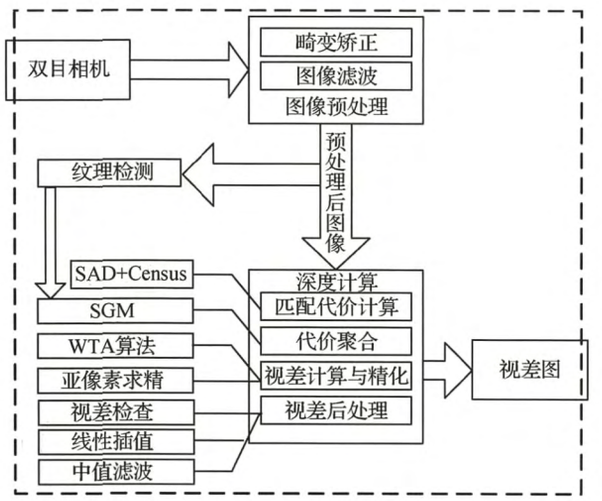


图 2 本文算法流程图

Fig. 2 The flow diagram of ADCC-TSGM algorithm

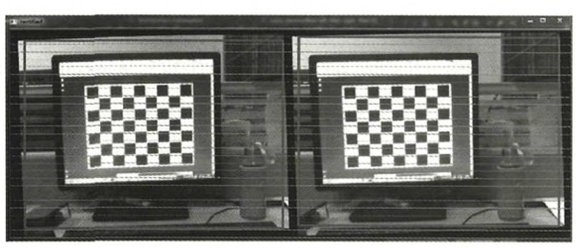


图 3 双目矫正图

Fig. 3 The stereo pairs after stereo rectification

1.2 匹配代价计算

本文使用中心平均 Census 特征及绝对误差(AD)特征计算匹配代价，使用均值作为匹配代价融合方式。AD 计算公式为

$$C_{\mathrm{AD}}(x,y,d) = \frac{1}{3}\sum_{c=\mathrm{R,G,B}} |I_L(x,y,c) - I_R(x-d,y,c)| \quad (1)$$

式中，$C_{\mathrm{AD}}(x,y,d)$ 为像素 (x,y) 在视差为 d 时的代价值。通过计算 AD 代价计算可以简单地处理连续变化区域的视差。

Census 特征则是一种非线性纹理特征。其将某一固定窗口中的像素以 Census 变换的方式进行二进制编码。通过计算两个 Census 特征的汉明距离得到可得到匹配代价值。

Census 特征变换首先设定一个窗口 W，并将 W 中的像素以固定顺序标号。以窗口中心像素 p 为基准，如果窗口中的一个标号为 i 的像素的灰度值大于中心像素，则将该位置为 1，否则置为 0。最终将生成的 01 矩阵组合为二进制串，完成变换。Census 变换定义为

$$Census(p) = \sum_{i=0}^{N} s(I_{(p)} - I_{(i)})2^i,\ s(x) = \begin{cases} 1 & x > 0 \\ 0 & x \leqslant 0 \end{cases} \quad (2)$$

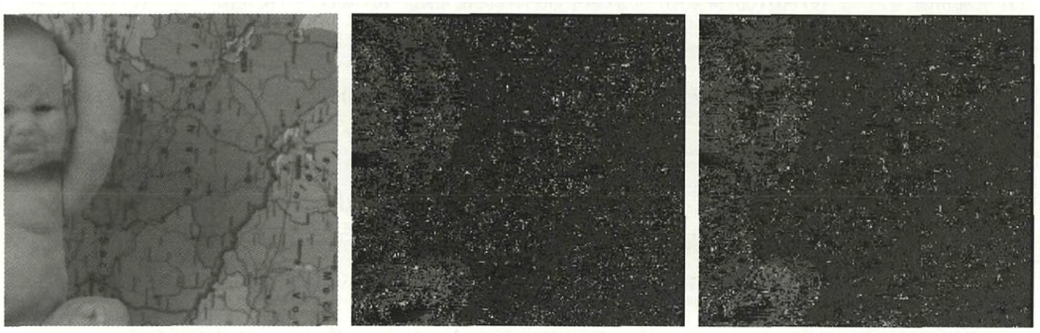
由式(2)可知,Census 特征变换本质上是以中心像素为基准对图像块进行二值化处理,因而中心像素 p 的大小对特征结果有较大影响。因此为提高 Census 特征鲁棒性,本文提出了中心平均 Census 特征(CA-Census)为

$$CA_{\text{Census}}(p) = \sum_{i=0}^{N} s(I_{\text{CA}(p)} - I_{(i)})2^i, s(x) = \begin{cases} 1 & x > 0 \\ 0 & x \leqslant 0 \end{cases} \quad (3)$$

$$I_{\text{CA}}(x,y) = \frac{\sum_{i=-1}^{i} I(x-i,y) + \sum_{i=-1}^{i} I(x,y+i)}{6} \quad (4)$$

经过这样的处理后,图像块中心的加权平均替换中心像素参与计算,Census 变换结果的鲁棒性得以提升。使用 CA-Census 进行逐点进行视差求取的效果如图 4 所示,可见使用后的图 4(c)在右侧地图区域白色噪点明显少于图 4(b)。

同时,为充分利用存储空间,加快处理速度,本文将编码窗口设定为 7×5 大小,并剔除四周较远的像素点。这样编码的 CA-Census 特征可以存入单个 32 位整形数字中,后续计算仅需异或操作和计数即可。如图 5 所示。



(a) 左视角图　(b) Census 效果图　(c) CA-Census 效果图

图 4　匹配代价计算效果对比

Fig. 4　The disparity map results of matching cost computation

((a) leftview; (b) using Census feature; (c) using CA-Census feature)

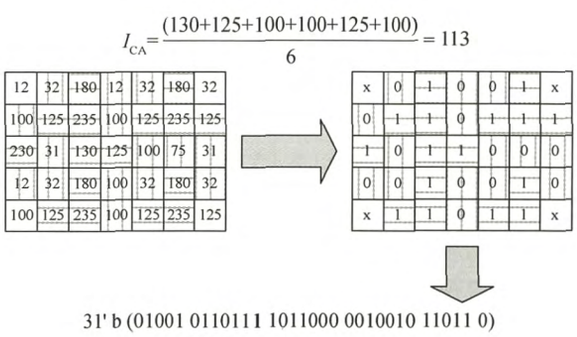


图 5　CA-Census 编码示意图

Fig. 5　The coding flow diagram of CA-Census feature

经 CA-Census 变换后,即可通过异或操作计算左视角图 $CA^{\text{l}}_{\text{Census}}(x,y)$ 与右视角图 $CA^{\text{r}}_{\text{Census}}(x,y)$ 之间的汉明距离 $Hamming(x,y,d)$ 得到 CA-Census 代价 $C_{\text{CA-Census}}(x,y,d)$ 为

$$Hamming(x,y,d) = CA^{\text{l}}_{\text{Census}}(x,y) \oplus CA^{\text{r}}_{\text{Census}}(x,y-d) \quad (5)$$

$$C_{\text{CA-Census}}(x,y,d) = Hamming(x,y,d) << 3 \quad (6)$$

通过将海明距离 $Hamming(x,y,d)$ 左移 3 位,可将 AD 代价和 CA-Census 代价缩放至[0,256)区间,便于继续处理。

考虑到 CA-Census 特征为周围像素点与中心像素差分二值化产生,与局部二值特征(LBP)相似程度极高。参考 LBP 的性质,不难发现其优势在于对有纹理的区域进行匹配,而对具体纹理差分值的大小并不敏感。由于其具有灰度不变性,对于线性光照等外部干扰有很好的鲁棒性。另一方面,由于本文所使用的 AD 匹配未进行窗口聚合,所以对于连续变化的低纹理区域,使用 AD 匹配可获得较高精度,但在低纹理区域极易造成无匹配的情况发生。因此,考虑到计算速度及实现便利性,直接将 AD 匹配代价和 CA-Census 匹配代价线性叠加,即可获得一定提升,即

$$C_{\text{Fusion}}(x,y,d) = \frac{C_{\text{AD}}(x,y,d) + C_{\text{CA-Census}}(x,y,d)}{2} \tag{7}$$

这样的聚合方式保证了在高纹理区域中，CA-Census 匹配代价占有较大的比重，而在高变化区域中 AD 匹配代价也能得到充分利用。使用融合后代价逐点进行视差求取的效果对比如图 6 所示。

(a) 左视角图　(b) AD代价视差图　(c) CA-Census代价视差图　(d) 融合代价视差图　(e) 真实视差图

图 6　代价融合结果对比示意图

Fig. 6　The disparity map results using pixel matching cost after feature fusion

((a) left view; (b) using AD feature; (c) using CA-Census feature; (d) using AD-CA-Census feature; (e) ground truth)

1.3　SGM 代价聚合

本文使用基于纹理优化的半全局匹配算法(SGM)[13]对匹配代价进行聚合，提高了低纹理区域匹配精度。

1.3.1　SGM 算法

SGM[13]作为动态规划(DP)算法的改进，极大地提升了算法效果。SGM 与大多数全局匹配算法相同，通过优化一个如下所示的凸能量函数，使其取得最小值，求取各个像素视差。

$$E(D) = \sum_p C(p,D_p) + \sum_{q\in N_p} P_1 T(|D_p - D_q| = 1) + \sum_{q\in N_p} P_2 T(|D_p - D_q| > 1) \tag{8}$$

$$T(x) = \begin{cases} 1 & x \text{ 成立} \\ 0 & x \text{ 不成立} \end{cases} \tag{9}$$

式中，$E(D)$ 为视差图 D 的能量函数，$C(p,D_p)$ 为 p 处的匹配代价，q 为 p 的邻域 N_p 中的像素，D_p、D_q 分别为视差图 D 中像素 p、像素 q 的视差。考虑到立体视觉的连续性约束性质，在视差相近的情况下，添加一个较小的惩罚项 P_1，而视差较远时，添加一个较大的惩罚项 P_2。

在实际操作时，首先与传统的 DP 算法相同，构建单一方向的状态转移方程

$$L_r(p,d) = C(p,d) + \min(L_r(p-r,d), L_r(p-r,d+1) + P_1, L_r(p-r,d-1) + P_1, \min_i(L_r(p-r,i)) + P_2) - \min_k(L_r(p-r,k)) \tag{10}$$

式中，方向 L_r 像素 p 视差为 d 的代价 $L_r(p,d)$ 由匹配代价 $C(p,d)$ 及上一状态 $L_r(p-r,d)$ 求得。通过在单一像素的所有视差减去相同值 $\min_k(L_r(p-r,k))$，可减少数据存储。

之后使用扫描线优化(scan line optimization)，将 8 个扫描线上的 L_r 相加，求得像素 p 视差 d 的聚合后代价为

$$S(p,d) = \sum_r L_r(p,d) \tag{11}$$

通过利用赢家通吃(WTA)算法，使用扫描线优化前后的聚合代价求解图 6(a)所示图像，可得如图 7 所示结果。其中图 7(a)—(h)是 8 个单一方向的 DP 所得视差图，图 7(i)(j)是扫描线优化后所得视差图及真实结果。可见通过扫描线优化后的 SGM 算法性能较单一方向 DP 算法有较大提升。

1.3.2　TSGM: 基于纹理优化的 SGM 改进算法

对于双目匹配而言，低纹理区域的匹配是性能提升的关键。因此在式(10)(11)的基础上，对 SGM 算法进行了改进，提出了一种基于纹理信息优化的半全局匹配(TSGM)。

使用灰度梯度作为纹理信息。对图像提取垂直方向和水平方向的梯度，可观察到如图8所示的实

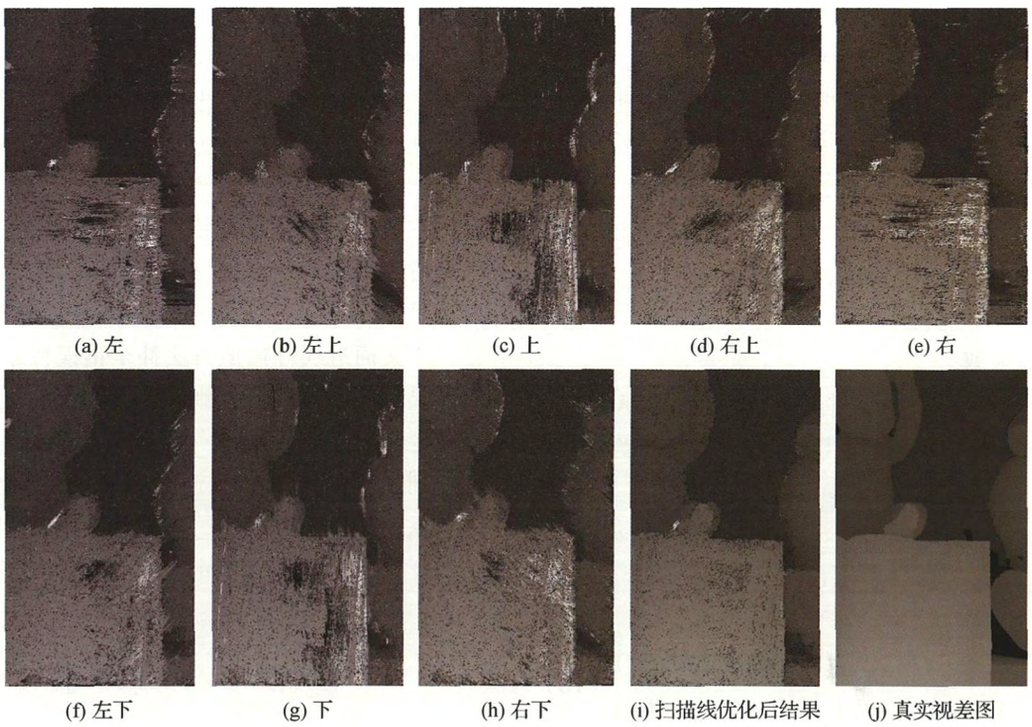


图 7　扫描线优化结果图

Fig. 7　The disparity map after scanline optimization

((a—h) the disparity map of eight different directions; (i) the disparity map after scanline optimization; (j) ground truth)

验图。在图 8(a) 中,粉色框为垂直方向的低纹理区域,蓝色框为水平方向低纹理区域。在图 8(b) 的视差图中,粉色区域的效果明显好于蓝色区域。这是由于实验所使用的图像是水平放置的双目摄像头所拍摄的,根据极线约束条件,匹配代价计算是沿极线方向进行的。从图 8(c) 和图 8(d) 中可以清楚地看到,在垂直方向上的低梯度区域(粉色部分) 恰好是水平方向上梯度值较高的区域,这些区域算法能较好地进行处理。而水平方向上的低梯度区域(蓝色部分) 则有较大的缺损出现。

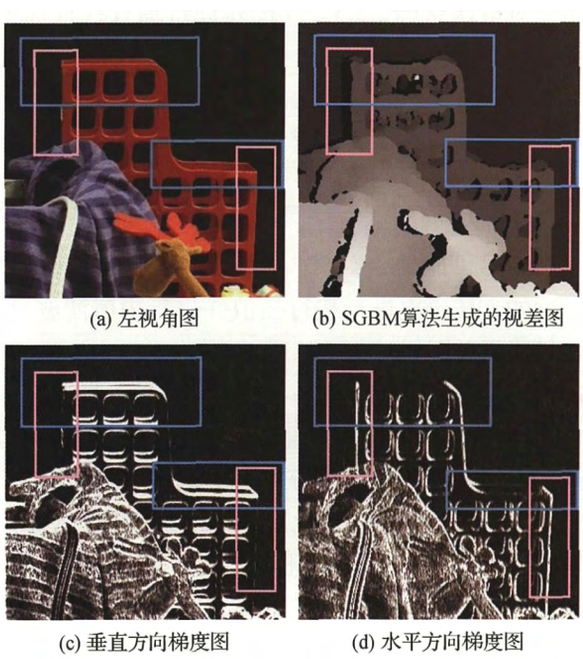


图 8　纹理信息示意图

Fig. 8　Thetexture information of left view

((a) left view; (b) disparity map generated by SGBM; (c) vertical gradient map; (d) horizontal gradient map)

因此,本文将水平方向的梯度作为纹理信息,通过在水平方向上进行 1 维差分计算,累加后得到纹理信息

$$texture_{(x,y)} = \sum_{x\in W_{(x)}}\sum_{y\in W_{(y)}} \mathrm{abs}(I_{(x+1,y)} - I_{(x,y)}) \tag{12}$$

式中,$I(x,y)$ 指在左视图点(x,y) 处的灰度值,abs 指绝对值函数,$texture(x,y)$ 指最后得到的纹理值,窗口 W 大小设置为 CA-Census 窗口大小,以确保此梯度信息能表征整个匹配代价窗口中的纹理信息。

当获取得到纹理信息后,可将其用于 SGM 代价

聚合之中。考虑到通常而言，低纹理区域视差平滑的可能性远高于高纹理区域，所以在式(10)引入约束以扩大低纹理区域出现视差变化的惩罚，即

$$\begin{aligned} P_1 &= P_{1_{\text{const}}} + \varepsilon_1 \times (255 - texture(p)) \\ P_2 &= P_{2_{\text{const}}} + \varepsilon_2 \times (255 - texture(p)) \end{aligned} \tag{13}$$

式中，ε_1、ε_2分别为 P_1、P_2的纹理惩罚系数，其中 P_1是视差变化为 1 的惩罚项，P_2是视差变化大于 1 的惩罚项。

1.4 视差求取

代价聚合后，使用改进的赢家通吃(WTA)算法得到粗略的视差值。此 WTA 算法计算公式为

$$d_{m1}(x,y) = \arg\min(S(x,y,d)) \tag{14}$$

$$d_{m2}(x,y) = \arg\min(S(x,y,d)),\ d \in [0, d_{\max}) \text{ 且 } d \neq d_{m1}(x,y,d) \tag{15}$$

$$d(x,y) = \begin{cases} 0 & S(x,y,d_{m1}) \geqslant \varepsilon \times S(x,y,d_{m2}) \\ d_{m1}(x,y) & S(x,y,d_{m1}) < \varepsilon \times S(x,y,d_{m2}) \end{cases} \tag{16}$$

式中，$d_{m1}(x,y)$表示最小代价视差值，$d_{m2}(x,y)$表示次小代价视差值，$S(x,y,d)$为 SGM 聚合后的代价值，d_{max}表示计算时的最大视差，$d(x,y)$为点(x,y)处的粗略视差。

参数 ε 用于判断视差计算是否有效，若最小代价 $S(x,y,d_{m1})$ 大于次小代价 $S(x,y,d_{m2}) \times \varepsilon$，则可视作此最小代价是不稳定的，将其置 0 表示视差计算错误。

计算完粗略的视差值后，使用二次拟合进行亚像素求精。令 d 表示求得的粗略视差，$d_- = d - 1$，$d_+ = d + 1$，则可由二次函数 $f(x) = ax^2 + bx + c$ 及其极值点求解公式 $x_{\min} = -b/2a$ 可知，拟合最佳视差为

$$d_{\min} = d - \frac{f(d_+) - f(d_-)}{2(f(d_+) + f(d_-) - 2f(d))} \tag{17}$$

式中，$f(d)$代表聚合后的代价函数，即 $S(x,y,d)$。

1.5 视差后处理

经过视差求取后，视差仍存在大量错误及空洞，因此需要进一步进行视差检查和空洞填充。考虑到立体视觉中左右视图是同一场景，根据视差一致性约束，稳定点在水平方向上从左至右和从右至左匹配应是相同的，若两个方向匹配不一致，则为不稳定或遮挡点。

设 D_{Ml}与 D_{Mr}分别表示左右视差图，$D_{Ml}(x,y)$与 $D_{Mr}(x,y)$分别表示左、右视差图在点(x,y)处的视差值，则稳定点的视差应满足

$$\begin{aligned} D_{Ml}(x,y) &= D_{Mr}(x - D_{Ml}(x,y), y) \\ D_{Mr}(x,y) &= D_{Ml}(x + D_{Mr}(x,y), y) \end{aligned} \tag{18}$$

不满足上述情况的点为不稳定或遮挡点，表明这些位置出现视差计算错误。若视差错误较小，则将左右视图的视差值进行插值，否则将其置为 0，视为错误视差。

本文通过线性插值方法补全错误视差。首先寻找错误点$(x_{\text{error}}, y_{\text{error}})$上侧视差 $D_M(x_{\text{error}}, y_{\text{top}})$大于 0 的稳定点$(x_{\text{error}}, y_{\text{top}})$及下侧的稳定点$(x_{\text{error}}, y_{\text{bottom}})$。考虑到错误点周围的稳定点并不可靠，搜索区域为错误点周围 $D_{\max}/4$ 至 $D_{\max}/2$ 之间的区域，即

$$\begin{aligned} y_{\text{top}} &= \arg\min(y_{\text{error}} - y_{\text{top}}) \\ &\text{s.t. } D_{\text{M}}(x_{\text{error}}, y_{\text{top}}) > 0 \text{ 且} \\ &\frac{D_{\max}}{2} > y_{\text{error}} - y_{\text{top}} > \frac{D_{\max}}{4} \\ y_{\text{bottom}} &= \arg\min(y_{\text{bottom}} - y_{\text{error}}) \\ &\text{s.t. } D_{\text{M}}(x_{\text{error}}, y_{\text{bottom}}) > 0 \text{ 且} \\ &\frac{D_{\max}}{2} > y_{\text{bottom}} - y_{\text{error}} > \frac{D_{\max}}{4} \end{aligned} \tag{19}$$

式中，$D_{\max}$指最大视差。而后按照稳定点至错误点距离对错误点$(x_{\text{error}}, y_{\text{error}})$按列进行加权线性插值得到修复视差图

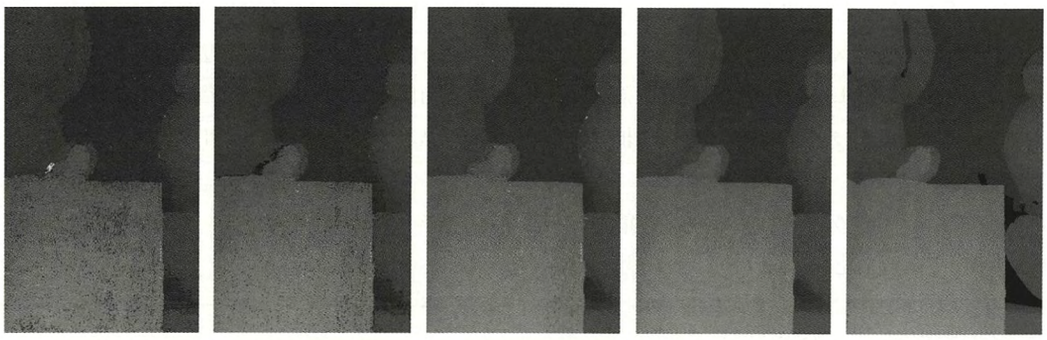
$$D_{\text{MF}}(x_{\text{error}}, y_{\text{error}}) = \frac{\left[\begin{matrix} D_{\text{M}}(x_{\text{error}}, y_{\text{top}}) \times (y_{\text{error}} - y_{\text{top}}) + \\ D_{\text{M}}(x_{\text{error}}, y_{\text{bottom}}) \times (y_{\text{bottom}} - y_{\text{error}}) \end{matrix}\right]}{y_{\text{bottom}} - y_{\text{top}}} \tag{20}$$

为尽可能减少空洞区域，在按列插值完成后以相同的方法再次进行按行插值并进行中值滤波，得出精确视差图，如图 9(d)所示。

1.6 CUDA 并行优化

上述算法虽对立体匹配算法有所改进，然而对于微小飞行器实时避障等实际应用而言，算法的运行速度至关重要。为此，本文通过 NVIDIA CUDA[14-15] 对算法进行了优化以实现并行加速。

在 NVIDIA 的 GPU 内存体系结构中，每个线程拥有独立的寄存器(registers)，线程块共享一组共享内存(shared memory)，而任意线程均可访问全局内存(gloal memory)[16]。各个等级内存的访问速度有明显差异，共享内存的访问延迟比全局内存低数十倍，而寄存器访问速度又较共享内存更快。



(a) 未经处理视差图　(b) 视差检查后视差图　(c) 线性插值视差图　(d) 中值滤波视差图　(e) 真实视差图

图 9　视差后处理优化结果图

Fig. 9　The disparity map after post-processing((a) before post-processing; (b) after left-right consistency check; (c) after linear interpolation; (d) after median filter; (e) ground truth)

对于如 CA-Census 特征计算这类访存复杂度远高于计算复杂度、而在相邻线程间又有大量的数据关联性的数据密集型计算任务,使用共享内存有利于提升计算速度。如图 10 中所示的线程格(Grid) 中拥有 21 个线程块(block) ,每个线程块包含 7 × 7 个线程。这些线程所处理像素的邻域是相互重叠的。因此可将整个线程块所需的数据分 4 个区域拷贝入共享内存,再使用共享内存计算 Census 特征,如算法 1 所示。

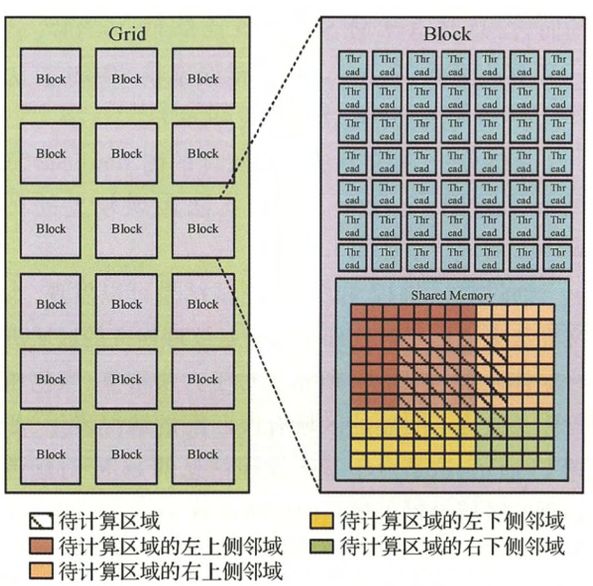


图 10　共享内存优化示意图

Fig. 10　The shared memory optimization diagram

census 共享内存化算法

function　CENSUS _ KERNEL (*src*, *dst*, *width*, *height*, *x*, *y*)

/* src_s为共享内存* /

if *x*,*y* 未越界 then

/* 将原数据块左上侧拷贝至共享内存* /

src_s [*top_left*] = *src* [*top_left*]

end if

if *x*,*y* 未越界 then

/* 将原数据块右上侧拷贝至共享内存* /

src_s [*top_right*] = *src* [*top_right*]

end if

/* 拷贝左下及右下侧数据* /

. . . .

/* 同步线程块内线程* /

syncthreads( )

/* 计算 census* /

dst [*y* * *width* + *x*] = CENSUS(src_s, *x*, *y*)

end function

function　CENSUS _ HOST (*src*, *dst*, *width*, *height*)

parallel for *block* in *grid* do

parallel for *thread* in *block* do

x = *thread. x* + *block. x* * *thread. xsize*

y = *thread. y* + *block. y* * *thread. ysize*

/* 调用 census 核函数* /

CENSUS_KERNEL(*src*, *dst*, *x*, *y*)

end parallel for

end parallel for

end function

NVIDIA 提供了大量 32 位 SIMD 指令以进一步加速计算。为了提高计算速度,在 SGM 计算中,单个

线程利用 SIMD 指令同时处理两个连续的视差值，并使用 shuffle 操作减少对全局内存的访问[17]。

在 GPU 处理期间，CPU 基本处于闲置状态。因此本文设计了一种混合流水线，以充分利用嵌入式平台的计算能力。

将双目立体匹配算法分割为预处理、代价计算、SGM 计算、视差计算、后处理五部分。预处理及后处理中的中值滤波使用 CPU 进行处理，而代价计算、视差计算及后处理中的其他部分由 GPU 完成，如图 11 所示。

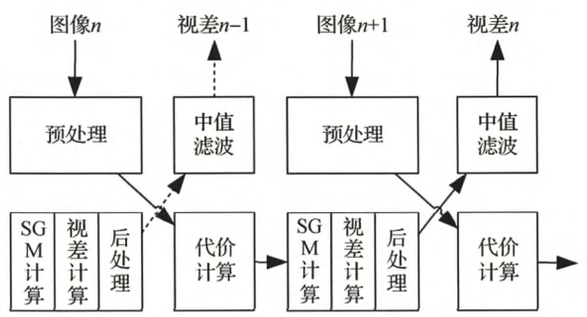


图 11　混合流水线优化示意图

Fig. 11　The hybrid pipeline optimization diagram

2　实验结果

2.1　算法性能

本文使用 Jetson TK1[18] 嵌入式计算平台进行算法测试，该平台搭载了四核 2.32 GHz Cortex-A15 架构的 ARM 处理器、192 个 Kepler 架构运算核心的 Tegra K1 GPU 以及 2 GB DDR3L 内存。测试集采用 (http://vision.middlebury.edu/stereo/data/scenes2006/)[19-20] 中的半尺寸图像数据集，对比算法采用 OPENCV 中自带的 BM 算法及 SGBM 算法，以及 SGM 算法[13]。由于 MC-CNN[7-8] 等一系列高性能算法所需内存过大，难以在此平台上部署，所以没有加入对比。由于在 CUDA 实现的 TSGM 算法经过高度优化，使用了 shuffle 等特殊处理，无法动态调节最大视差。考虑到实际应用场景，将各个算法的最大视差均设置为 64 像素，数据集分辨率为 QVGA 水平，SGBM 及 BM 算法窗口大小设置为 7×7，ADCC-TSGM 算法中纹理惩罚系数 ε_1 设置为 0.25，ε_2 设置为 0.125，视差有效性阈值 ε 设置为 0.95。

算法结果以左视差图为例，通过与真实视差图比较，计算总坏点率和平均错误率，进行性能评价。因为各个算法均未考虑边界遮挡区的影响，所以边界遮挡区不计入评价标准中。

总坏点率为坏点率和无效点率之和为

$$\delta_{\text{total}}=\delta_{\text{bad}}+\delta_{\text{invalid}} \tag{21}$$

坏点率

$$\delta_{\text{bad}}=\frac{\sum_{p\in I}\left(\left|Disp_{\text{C}}(p)-Disp_{\text{GT}}\right|>\delta_d\right)}{N_{\text{GT}}} \tag{22}$$

式中，$Disp_{\text{C}}$ 指算法求得的视差图，$Disp_{\text{GT}}$ 指真实视差图，N 为真实视差图的有效像素总数，δ_d 为视差错误容差。考虑到微小飞行器避障的实际需求，将 δ_d 设置为 4。

无效点率

$$\delta_{\text{invalid}}=\frac{\sum_{p\in I}\left(Disp_{\text{C}}(p)=0\right)}{N_{\text{GT}}} \tag{23}$$

平均错误率

$$\delta_{\text{average}}=\frac{\sum_{p\in I}\left(\left|Disp_{\text{C}}(p)-Disp_{\text{GT}}\right|\right)}{N_{\text{GT}}} \tag{24}$$

通过测试数据集中 21 幅图像，分析不同算法之间的差异。如表 1 所示，大多数情况下，本文算法 (ADCC-TSGM) 均优于 BM、SGBM 及 SGM 算法。平均而言，总坏点率较 SGBM 算法降低 36.1%，较 SGM 算法降低 28.3%。平均错误率较 SGBM 算法降低 44.5%，较 SGM 算法降低 49.9%。

具体分析，本文算法表现较优的图 Bowling2 测试结果如图 12 所示。本文算法表现较差的图 Wood1 测试结果如图 13 所示。

通过对比图 12 和图 13 测试结果可以发现，本文算法更适用于对纹理丰富或是纹理连续的区域进行立体匹配，如图 12 所示。对于大块重复、颜色相近而又纹理不明显的区域有误匹配的情况发生，如图 13 所示，这是由于基于纹理信息的 SGM 对颜色相近的区域设置了较高的惩罚系数。

此外，将本文算法所改进的各个部分进行精简或替换后进行测试，如表 2 所示。由表 2 可知，本文所做的改进能有效提高算法性能。具体而言，使用 CA-CENSUS 进行特征匹配降低了坏点率和错误率。在 SGM 中使用纹理信息能有效降低平均错误率。后处理中的视差检查和空洞补全可大幅降低坏点率和错误率。

表 1　不同算法的总坏点率与平均错误率

Table 1　Total bad pixel rate and average error rate of different stereo algorithms

图像	BM		SGBM		SGM [13]		**本文算法**	
	δ_{total}	$\delta_{average}$	δ_{total}	$\delta_{average}$	δ_{total}	$\delta_{average}$	δ_{total}	$\delta_{average}$
Aloe	28.84	22.44	16.84	11.73	**9.01**	8.23	9.39	**6.16**
Baby1	14.98	11.86	8.13	5.35	**4.64**	9.30	5.61	**3.45**
Baby2	19.37	18.45	5.79	3.89	3.50	5.60	**2.77**	**2.29**
Baby3	18.55	21.53	7.78	6.71	7.42	6.07	**5.04**	**3.47**
Bowling1	37.23	48.67	10.5	9.24	8.78	7.26	**7.01**	**6.18**
Bowling2	20.58	27.38	7.08	7.22	7.25	13.34	**4.74**	**3.83**
Cloth1	5.75	5.93	1.43	1.86	0.34	4.13	**0.16**	**0.85**
Cloth2	13.47	18.76	5.44	6.02	**3.32**	9.13	3.81	**3.34**
Cloth3	9.27	9.92	1.74	1.96	1.91	4.81	**0.78**	**1.33**
Cloth4	14.97	15.84	7.86	6.45	**3.22**	**3.19**	5.80	5.22
Flowerpots	21.44	27.92	5.11	5.06	4.83	9.39	**3.66**	**3.11**
Lampshade1	43.76	39.73	15.22	9.8	10.31	15.61	12.57	**7.48**
Lampshade2	45.71	45.58	14.98	10.32	23.29	18.44	**9.91**	**6.82**
Midd1	51.48	38.91	28.39	17.16	42.30	19.53	**25.26**	**11.40**
Midd2	52.32	33.45	27.72	13.8	26.01	13.30	**23.97**	**10.60**
Monopoly	35.60	22.37	19.36	11.84	17.75	15.08	**8.73**	**6.99**
Plastic	70.92	102.89	26.22	28.86	21.00	19.62	**8.15**	**6.00**
Rocks1	10.84	11.25	3.03	2.96	2.41	5.79	**1.53**	**1.87**
Rocks2	8.62	9.01	3.38	3.26	2.62	5.45	**1.98**	**1.89**
Wood1	25.7	33.38	11.49	11.77	**3.72**	**3.52**	6.86	6.82
Wood2	24.42	39.92	6.69	7.48	4.88	5.57	**1.86**	**2.42**
平均值	27.32	28.82	11.15	8.70	9.93	9.64	**7.12**	**4.83**

注: 粗体为最优结果。

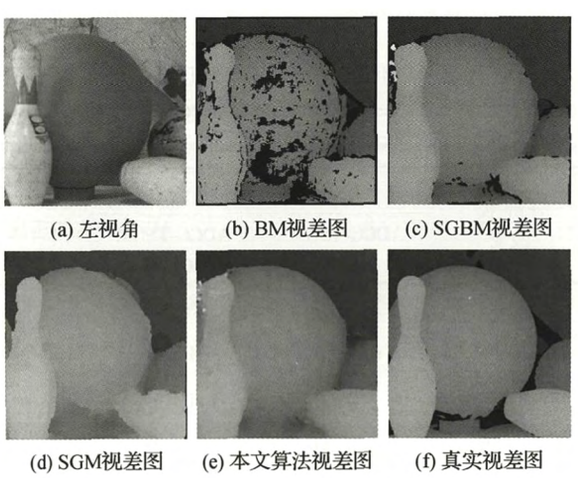


图 12　Bowling2 测试结果图

Fig. 12　The results of Bowling2((a) left view; (b) BM; (c) SGBM; (d) SGM; (e) SGBM; (f) ground truth)

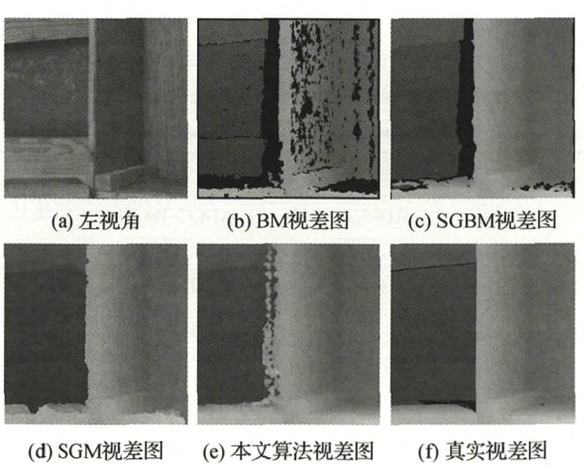


图 13　Wood1 测试结果图

Fig. 13　The results of Wood1((a) left view; (b) BM; (c) SGBM; (d) SGM; (e) SGBM; (f) ground truth)

表 2　算法精简后的总坏点率均值与平均错误率均值

Table 2　Average δ_{total} and $\delta_{average}$ of simplified algorithms

部分替换算法	δ_{total} 均值	$\delta_{average}$ 均值
CC-SGM	9.44	6.97
AD-SGM	17.48	10.20
ADCC-BM	12.11	9.29
ADCC-TSGM(固定惩罚系数)	8.92	6.62
ADCC-TSGM(无后处理)	14.46	11.49
SGM [13]	9.93	9.64
ADCC-TSGM	**7.12**	**4.84**

注: 粗体为最优结果。CC-SGM 指在代价计算中仅使用 CA-CENSUS 作为匹配代价。AD-SGM 指在代价计算中仅使用 AD 作为匹配代价。固定惩罚系数的 ADCC-TSGM 是指在代价聚合中不使用纹理信息。ADCC-BM 指使用 7×7 窗口替代 SGM 进行代价聚合。无后处理是指不进行左右视差检查和孔洞插值补全。

2.2　CUDA 并行优化

在 CUDA 并行优化实验中，各个算法的设置与 2.1 节相同。各个算法运行时间如图 14 所示。

由图 14 可见，使用 CUDA 加速后，算法的实时性得到了极大的提高，对 CPU 版本的平均加速比达到了 117.2 倍。即使相较于已进行 SIMD 及多核并行优化的 SGBM，运行时间也减少了 85%。

此外，如表 3 所示，通过比较算法各个部分的 CPU 版本和 GPU 版本耗时，可以发现，CUDA 并行加速可以有效地减少代价计算、SGM 聚合、视差求取阶段的耗时。

在不同分辨率下，CUDA 加速的算法在各个尺寸下均表现出良好的性能，加速比也较为稳定，几乎不随图像尺寸变化。值得注意的是，在 240×320 分辨率下，CUDA 加速后的 ADCC-TSGM 耗时仅 31.44 ms，运行速度为 31.8 帧/s，可满足嵌入式实时双目视觉需求。

由于代价计算、SGM 聚合及后处理在 CUDA 并行实现与 CPU 版本实现中存在细节差异，本文进而

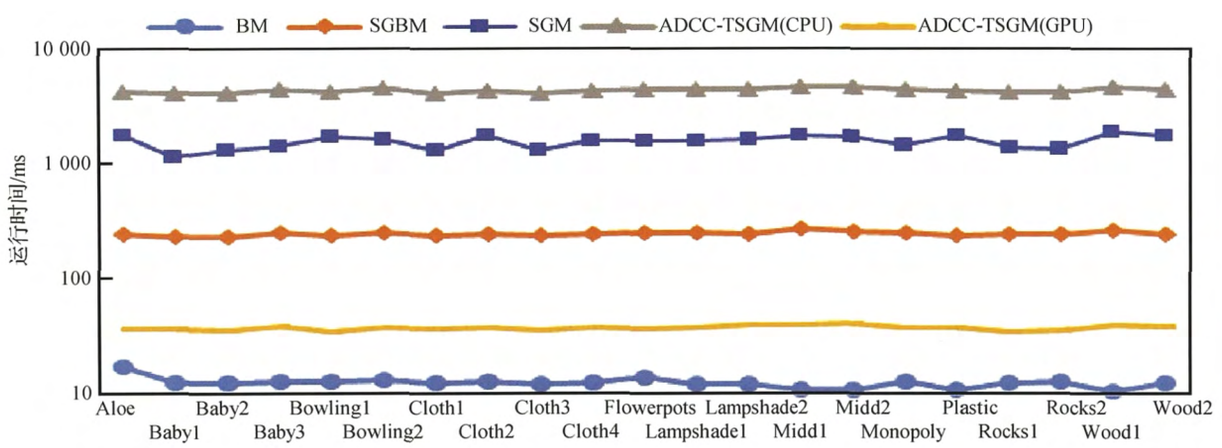


图 14　算法运行时间示意图

Fig. 14　Comparison of the running time of the algorithms

表 3　算法各部分运行时间与加速比

Table 3　Running time and speedup ratios of the parts in ADCC_SGM

	耗时/ms		加速比
	ADCC-TSGM (CPU)	ADCC-TSGM (GPU)	
代价计算	1 169.7	7.57	154.5
纹理信息计算	3.94	1.94	2.03
SGM 聚合	3 013.83	21.18	142.3
视差求取	97.38	3.77	25.83
视差后处理 (无中值滤波)	3.33	0.66	5.045
总计	4 288.18	35.11	122.13

注: 此处加速比略高于上文，因为此处未考虑中值滤波和预处理。中值滤波和预处理在 CPU 上执行，未有加速。

表 4　不同分辨率下耗时与加速比

Table 4　The run-time and speedup ratios of ADCC-TSGM related to image resolutions

分辨率/像素	耗时/ms		加速比
	ADCC-TSGM (CPU)	ADCC-TSGM (GPU)	
120×160	716.50	6.10	117.47
240×320	3 655.90	31.44	116.27
480×640	16 177.60	141.16	114.61
960×1 280	68 306.70	590.29	115.72

测试两种不同实现方式对性能的影响。由表 5 可知,两者的坏点率和平均误差率差异极小,平均小于 1%。即使对于坏点率和平均误差率差异相对较大的 Bowling1 图,实际的计算效果差别微乎其微,如图 15 所示。

表 5　不同实现方式的总坏点率与平均错误率

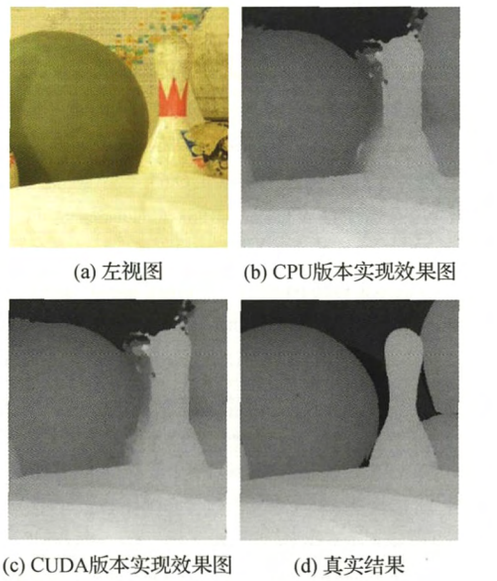
Table 5　Total bad pixel rates and average error rates of the CPU/GPU implementations of ADCC-TSGM

Image	ADCC-TSGM(CPU)		ADCC-TSGM(GPU)	
	δ_{total}	$\delta_{average}$	δ_{total}	$\delta_{average}$
Aloe	9.39	6.16	**9.29**	**6.10**
Baby1	5.61	**3.45**	5.61	3.49
Baby2	2.77	2.29	**2.74**	**2.28**
Baby3	5.04	**3.47**	**4.98**	3.48
Bowling1	7.01	6.18	**6.79**	**5.98**
Bowling2	**4.74**	3.83	4.76	**3.80**
Cloth1	0.16	**0.85**	**0.15**	0.87
Cloth2	**3.81**	**3.34**	3.83	3.37
Cloth3	**0.78**	**1.33**	0.78	1.35
Cloth4	5.80	**5.22**	**5.72**	5.35
Flowerpots	3.66	**3.11**	**3.62**	3.15
Lampshade1	12.57	7.48	**12.37**	**7.46**
Lampshade2	**9.91**	**6.82**	10.19	7.10
Midd1	25.26	11.40	**24.49**	**11.16**
Midd2	23.97	10.60	**23.62**	**10.56**
Monopoly	8.73	6.99	**8.52**	**6.92**
Plastic	**8.15**	**6.00**	9.16	6.70
Rocks1	**1.53**	**1.87**	1.55	1.89
Rocks2	**1.98**	**1.89**	2.01	1.92
Wood1	6.86	6.82	**6.44**	**6.42**
Wood2	**1.86**	**2.42**	1.93	2.52
平均值	7.12	**4.84**	**7.07**	4.85

注:粗体为最优结果。

3　结　论

本文提出了一种基于中心平均 census (CA-Census) 特征提取及纹理优化的改进 SGM 立体匹配算法——ADCC-TSGM 算法。相较于 OpenCV 中的 BM 及 SGBM 算法,ADCC-TSGM 算法的总坏点率与平均错误率均有明显下降。



(a) 左视图　(b) CPU版本实现效果图

(c) CUDA版本实现效果图　(d) 真实结果

图 15　CUDA 并行实现与 CPU 版本实现效果图

Fig. 15　The disparity maps generated by the CPU/GPU implementations of ADCC-TSGM

((a) left view; (b) CPU version of ADCC-TSGM; (c) CUDA version of ADCC-TSGM; (d) ground truth)

基于 CUDA 架构,本文在 NVIDIA TK1 嵌入式计算平台上对 ADCC-TSGM 算法进行了并行加速优化,在 QVGA 图像分辨率下获得了 117.2 倍的加速比,帧率可达 31.8 帧/s,可基本满足微小飞行器深度信息获取等实时性应用的要求。

本文虽融合了纹理和颜色两种匹配特征并在半全局匹配中引入纹理特征优化,但仍无法在大块重复、颜色相近而又纹理不明显的区域产生准确的视差图。在未来的工作中,将针对此问题尝试使用层次化和多尺度匹配的方式进行研究。

参考文献(References)

[1] Scharstein D, Szeliski R. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms [J]. International Journal of Computer Vision, 2002, 47(1-3): 7-42. [DOI: 10.1023/A:1014573219977]

[2] Kim J, Kolmogorov V, Zabih R. Visual correspondence using energy minimization and mutual information [C]//Proceedings of 2003 Ninth IEEE International Conference on Computer Vision. Nice, France: IEEE, 2003: 1033-1040. [DOI: 10.1109/iccv.

2003.1238463]

[3] Muquit M A, Shibahara T, Aoki T. A high-accuracy passive 3D measurement system using phase-based image matching [J]. IEICE Transactions on fundamentals of Electronics, Communications and Computer Sciences, 2006, E89-A(3): 686-697. [DOI: 10.1093/ietfec/e89-a.3.686]

[4] Roy S, Cox I J. A maximum-flow formulation of the N-camera stereo correspondence problem [C]//Proceedings of 1998 Sixth International Conference on Computer Vision. Bombay, India: IEEE, 1998: 492-499. [DOI: 10.1109/iccv.1998.710763]

[5] Zhang K, Lu J B, Lafruit G. Cross-based local stereo matching using orthogonal integral images [J]. IEEE Transactions on Circuits and Systems for Video Technology, 2009, 19(7): 1073-1079. [DOI: 10.1109/tcsvt.2009.2020478]

[6] Mei X, Sun X, Zhou M C, 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). Barcelona, Spain: IEEE, 2011: 467-474. [DOI: 10.1109/iccvw.2011.6130280]

[7] Žbontar J, LeCun Y. Stereo matching by training a convolutional neural network to compare image patches [J]. Journal of Machine Learning Research, 2016, 17(65) 1-32.

[8] Li L C, Yu X, Zhang S L, et al. 3D cost aggregation with multiple minimum spanning trees for stereo matching [J]. Applied Optics, 2017, 56(12): 3411-3420. [DOI: 10.1364/AO.56.003411]

[9] Hrabar S, Sukhatme G S, Corke P, et al. Combined optic-flow and stereo-based navigation of urban canyons for a UAV [C]//Proceedings of 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems. Edmonton, Canada: IEEE, 2005: 3309-3316. [DOI: 10.1109/iros.2005.1544998]

[10] Stefanik K V, Gassaway J C, Kochersberger K, et al. UAV-based stereo vision for rapid aerial terrain mapping [J]. GIScience & Remote Sensing, 2011, 48(1): 24-49. [DOI: 10.2747/1548-1603.48.1.24]

[11] Zhou G, Fang L, Tang K T, et al. Guidance: A visual sensing platform for robotic applications [C]//Proceedings of 2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops. Boston, USA: IEEE, 2015: 9-14. [DOI: 10.1109/cvprw.2015.7301360]

[12] Zhang Z. A flexible new technique for camera calibration [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2000, 22(11): 1330-1334. [DOI: 10.1109/34.888718]

[13] Hirschmuller H. Stereo processing by semiglobal matching and mutual information [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2008, 30(2): 328-341. [DOI: 10.1109/tpami.2007.1166]

[14] Nvidia C. Nvidia Cuda Compute Unified Device Architecture Programming Guide [M]. Santa Clara, USA: NVIDIA Corporation, 2007.

[15] Nvidia C. CUDA C Programming guide [EB/OL]. 2017-10-22 [2017-11-17]. http://docs.nvidia.com/cuda/cuda-c-programming-guide.

[16] The CUDA Memory Model. CUDA, Supercomputing for the Masses: Part 4 [EB/OL]. 2008-06 [2017-03-10]. http://www.drdobbs.com/parallel/cuda-supercomputing-for-the-masses-part/208401741?pgno=3.

[17] NvidiaKepler Compute Architecture. Tuning CUDA applications for kepler [EB/OL]. 2017-10-22 [2017-11-17]. http://docs.nvidia.com/cuda/kepler-tuning-guide.

[18] Nvidia J. Bringing GPU-accelerated computing to embedded systems [EB/OL]. 2014-04 [2017-03-10]. http://developer.download.nvidia.com/embedded/jetson/TK1/docs/Jetson_platform_brief_May2014.pdf.

[19] Scharstein D, Pal C. Learning conditional random fields for stereo [C]//Proceedings of 2007 IEEE Conference on Computer Vision and Pattern Recognition. Minneapolis, USA: IEEE, 2007: 1-8. [DOI: 10.1109/cvpr.2007.383191]

[20] Hirschmuller H, Scharstein D. Evaluation of cost functions for stereo matching [C]//Proceedings of 2007 IEEE Conference on Computer Vision and Pattern Recognition. Minneapolis, USA: IEEE, 2007: 1-8. [DOI: 10.1109/cvpr.2007.383248]