Flexible Hybrid Stereo Matching

Jingzhou Huang[1, a], Changyu Diao[2, b]

[1, 2]College of Computer Science and Technology, Zhejiang University

Hangzhou, China, 310027

[a]xiaosai567@gmail.com, [b]diaochangyu@gmail.com

Keywords: stereo matching, hybrid stereo, flexibility, extensibility

Abstract. We present a hybrid stereo matching algorithm, which decomposes the stereo matching process into a number of subtasks and deals with them using different methods. Our algorithm consists of three components: 1) segment the reference image into regions and regard each region as the basic task unit; 2) extract and match feature points of the stereo image pair; 3) determine the type of each task unit and handle it with a certain method. Tasks can run on CPU and GPU simultaneously, so different computing resources achieve a real cooperation. We test our algorithm on a PC with single CPU and single GPU, and results demonstrate its effectiveness.

Introduction

Stereo matching is an attractive topic in computer vision. Scharstein and Szeliski [1] have divided stereo methods into two categories, local and global methods, in their taxonomy. Local methods usually have a faster computing speed, while global ones are possible to obtain more accurate disparity maps. In recent years, the emergence of multi-core CPU and GPU has driven the parallelism of stereo algorithms, and it significantly accelerates matching process. However, most parallel stereo algorithms still work based on a particular computing resource and are lack of flexibility when they are deployed in systems with multiple computing resources. We study this issue and design a hybrid stereo algorithm for systems containing multiple computing resources, which has the following advantages: 1) our algorithm partitions the stereo process to multiple subtasks, which significantly reduces the peak requirement for memory, so that it is potential to support stereo matching using higher-resolution image pairs; 2) different types of tasks can be processed simultaneously on CPU and GPU, which improves the computational efficiency.

We organize our paper as follows: in next section we review the previous work; the proposed algorithm and its implementation are described in detail in the 3rd section; we test our algorithm on a PC with CPU and GPU, and present the experiment results in the 4th section; finally we conclude our work and demonstrate the future work in the 5th section.

Previous Work

Adaptive support-weight (ASW) algorithms currently represent the state of the art in local stereo matching because of their comparable accuracy with most global stereo methods. Hosni [2] has implemented nine ASW stereo methods and evaluated their performance from the view of accuracy and computational efficiency of their GPU implementations. Kowalczuk [3] also proposes an iterative refinement method for ASW and run it on GPU. Experiment results show the ASW based approach can achieve real-time processing speed and very high accuracy.

In global inference based stereo methods, Graph Cut [4] and Belief Propagation (BP) [5] are two most popular ones. Choi [6] has proposed a strategy suitable for GPU and achieved 5.2X speedup over the baseline implementation. In contrast, BP is easier to parallelize, therefore, BP is used more widely in parallel stereo methods [7].

Besides parallelizing existing methods, some researchers have presented new approaches to reduce the computing cost. Sinha [8] exploits sparse stereo correspondences to perform local plane

sweeps without exploring the full disparity search space, so his algorithm achieves significant speedup over previous ones and supports stereo matching with very high resolution image pairs.

Above accelerated algorithms are usually developed for a certain parallel platform without taking extensibility into consideration, so they cannot achieve a real concurrent execution between CPU and GPU.

Algorithm Description and Implementation

Our algorithm is made up of three main components: 1) feature point extraction and matching for the stereo image pair; 2) image segmentation; 3) task assignment and computation. The structure of our algorithm is given in Fig. 1 (a).

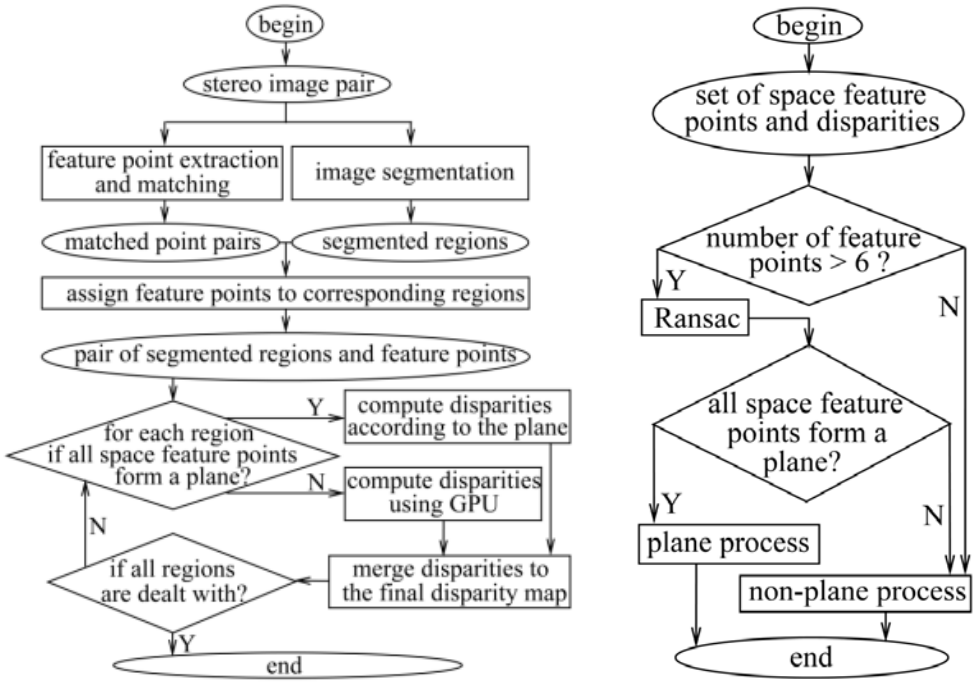


Fig. 1 (a) The structure of our algorithm; (b) Rules of determining if a piece of region forms a plane

Feature Points Extraction and Matching. Dense stereo matching has very expensive computation cost, so it is impractical if you use a big size of matching window or an stereo image pair with a high resolution. In contrast, sparse matching based on feature points has a lower computation complexity and a higher reliability, therefore we regard it as an auxiliary procedure to task decomposition and classification. There are no rotation and illumination changes for stereo matching, so SURF is a natural choice for us [9]. We adopt the GPU-SURF implementation in the OpenCV library to perform feature point extraction and matching.

Image Segmentation. The stereo matching process is decomposed according to segmentations. Here we employ an assumption: neighboring disparities in a segmentation vary smoothly or stay unchanged, and disparity saltation only appears on the edge of segmentations. In order to avoid resource conflict and implement the task parallelism with *feature points extraction and matching*, we adopt the technique in [10] to perform *image segmentation*.

Task Assignment and Computation. Tasks obtained according to segmented regions are classified as two types: planar and non-planar ones. This judgment is completed through feature points involved in the current region. We suggest rules in Fig. 1 (b) to classify tasks, and we employ RANSAC [11] to improve the robustness of estimation.

When classifying tasks, we have obtained a plane equation for each planar region. After a region has been judged as a plane, we can compute disparities of this region with the equation. While for a non-planar region, we still have to compute disparities point by point. Here we use the SAD to compute the initial cost, and loopy BP to optimize it. It is worth noting that the two approaches are

applied in regions but the whole image. In order to reach the task-level parallelization, we implement SAD and loopy BP on GPU.

Experiments

The window size is 17×17 for SAD and the iteration number is 5 for loopy BP in our algorithm, and they are same for four data sets on Middlebury benchmark. We count the computation time of each component and give the throughput for plane processing in Table 1:

Table 1. Computation time of each component and throughput for plane processing

	Tsukuba	Venus	Teddy	Cones
Resolution (W×H)	384×288	434×383	450×375	450×375
Plane percentage (%)	25.6	49	24.6	12.4
Feature points (ms)	36	48	45	57
Image seg (ms)	86	121	132	137
Subtasks assign (ms)	5	5	3	4
Plane computation (ms)	0.0983	0.2976	0.2299	0.0979
Plane computation speed (Mp/s)	288.011	280.277	180.568	213.739

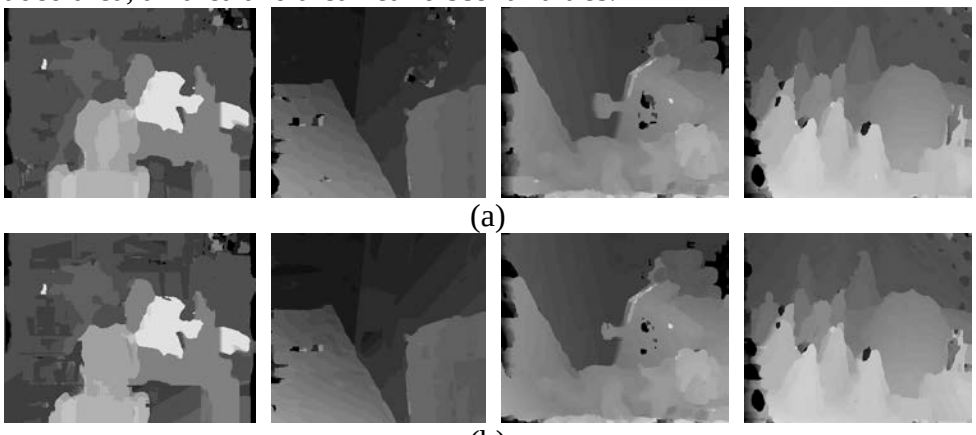
From the table we can see that all components have very high computation speed, and the average process speed is about 200Mpixels/s. Next we compare the computation speed and the peak memory loopy BP consumes in processing non-planar regions, and the data is put in Table 2.

Table 2. Comparison for average speed and peak memory consumption between loopy BP in our algorithm and loop BP on the CPU

Loopy BP	Tsukuba	Venus	Teddy	Cones
Speed on the GPU (Mp/s)	0.4714	0.3241	0.0673	0.0643
Average speed of our algorithm (Mp/s)	0.367	0.381	0.231	0.106
Average speed on the CPU (Mp/s)	0.3301	0.2743	0.1003	0.0974
Peak Memory on the GPU (MB)	6.4082	20.4101	30.1171	67.9883
Peak Memory on the CPU (MB)	61.3828	115.0867	348.5799	348.5799

For loopy BP implementation, the computation is performed region by region. From Table 2, we know compared with its CPU version, the peak memory requirement on the GPU is reduced significantly, and its computation speed is comparative with the CPU implementation.

We display generated disparity maps in Fig. 2 and put error rates in Table 3 respectively from CPU and our algorithm using Middlebury benchmark, where *n-o*, *all*, and *disc* indicate the percentage error for non-occluded area, all area and area near discontinuities.



(a)

(b)

Fig. 2 (a) Results based on SAD and loopy BP on the CPU; (b) Results from our algorithm with SAD and loopy BP.

Table 3. Comparison for error rates between CPU implementation and our algorithm

data set / error rate	tsukuba			venus			teddy			cones		
	n-o	all	disc	n-o	all	disc	n-o	all	disc	n-o	all	disc
CPU (%)	8.54	10.3	36.6	6.16	7.27	40.5	19.0	25.8	42.5	13.2	21.1	34.0
our algorithm (%)	8.59	10.0	34.7	9.36	10.3	40.6	20.2	25.8	41.0	14.4	22.2	34.6

Error rates show that our algorithm has a comparable accuracy with that of the corresponding SAD and loopy BP version on the CPU, that is, our approach does not cause obvious accuracy reduction. It verifies the feasibility of our algorithm.

Conclusion and Future Work

In this paper we propose an efficient hybrid stereo matching algorithm, which can achieve cooperation between different computation resources. The experiment results show the feasibility of our algorithm. Our future work will focus on exploring a dynamic load balancing strategy between different computing resources to boost overall efficiency.

Acknowledgement

This work is supported by Issue of National Science and Technology Support Plan (No. 2012BAH03F02, No. 2013BAK08B08), Important Project Subject to Tender of The National Social Science Fund (No. 12&ZD32), and Science and Technology Project on Heritage Conservation (Detection and Simulation Technology for Heritage Preservation Environment in Archaeological Sites), Zhejiang Province.

References

[1] D. Scharstein, R. Szeliski, A taxonomy and evaluation of dense two-frame stereo correspondence algorithms, Int. J. Comput. Vision 47 (2002) 7-42.

[2] A. Hosni, M. Gelautz, M. Bleyer, Accuracy-efficiency evaluation of adaptive support weight techniques for local stereo matching, Pattern Recognit. 7476 (2012) 337-346.

[3] J. Kowalczuk, E.T. Psota, L.C. Perez, Real-time stereo matching on CUDA using an iterative refinement method for adaptive support-weight correspondences, IEEE Trans. Circuits Syst. Video Technol. 23 (2013) 94-104.

[4] O. Veksler, Graph cut based optimization for MRFs with truncated convex priors, 2007 IEEE Conference on Computer Vision and Pattern Recognition 1-8 (2007) 2142-2149.

[5] P.F. Felzenszwalb, D.P. Huttenlocher, Efficient belief propagation for early vision, Int. J. Comput. Vision 70 (2006) 41-54.

[6] Y.K. Choi, I.K. Park, Efficient GPU-based graph cuts for stereo matching, 2013 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW) (2013) 642-648.

[7] A. Ahmadzadeh, H. Madani, K. Jafari, F.S. Jazi, S. Daneshpajouh, S. Gorgin, Fast and adaptive BP-based multi-core implementation for stereo matching. 2013 Eleventh IEEE/ACM International Conference on Formal Methods and Models for Codesign (2013) 135-138.

[8] S.N. Sinha, D. Scharstein, R. Szeliski, Efficient high-resolution stereo matching using local plane sweeps. 2014 IEEE Conference on Computer Vision and Pattern Recognition (2014) 1582-1589.

[9] L. Juan, O. Gwun, A comparison of SIFT, PCA-SIFT and SURF, International Journal of Image Processing 3 (2009) 143-152.

[10] P.F. Felzenszwalb, D.P. Huttenlocher, Efficient graph-based image segmentation, Int. J. Comput. Vision 59 (2004) 167-181.

[11] M.A. Fischler, R.C. Bolles, Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography, Communications of the ACM 24 (1981) 381-395.