CUDA-based GPU Implementation of Hierarchical Belief Propagation for Fast Stereo Matching

Jae-Il Jung and Yo-Sung Ho
Gwangju Institute of Science and Technology (GIST)
261 Cheomdan-gwagiro, Buk-gu, Gwangju, 500-712, Korea
Telephone: +82-62-715-2258, Fax: +82-62-715-3164
E-mail: {jijung, hoyo}@gist.ac.kr

***Abstract*- Stereo matching based on the Markov random field model has a global optimization problem. Solutions of the problem can be inferred by the belief propagation (BP) algorithm. The BP algorithm effectively estimates global solutions, but it takes a very long time to calculate messages. In this paper, we implement the hierarchical BP algorithm on a graphics processing unit (GPU) using compute unified device architecture. We perform memory uploads to the global memory of the GPU, and compute messages of all pixels in parallel. We analyze the performance according to block and image sizes. Experimental results show that our implementation is faster than the conventional loopy BP algorithm and the hierarchical BP algorithm implemented on CPU.**

I. INTRODUCTION

A depth image represents distances between a camera and objects. It is essential for the free-view television system, and various techniques have been developed to accurately estimate it [1]. Stereo matching is a popular technique for depth estimation, and is one of the most heavily investigated topics in computer vision [2].

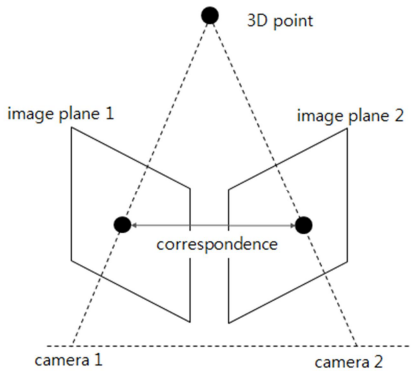


Figure 1. Stereo matching for depth estimation

As shown in Fig. 1, stereo matching finds correspondences between stereo pairs. With correspondences and camera centers, the real positions of the correspondences can be calculated. At the beginning of the research, the pixel- or block-based stereo matching algorithms whose costs include squared intensity differences and absolute intensity differences were used. Although they have low-computational complexity, their performance is not guaranteed. The performance significantly depends on textures of input stereo pairs. Depth image's quality decreases especially when the input pair contains many occlusion and texture-less regions.

One recent advance involves the use of Markov random field (MRF) models to improve the performance in the mentioned regions. MRF is a graphical model in which a set of random variables have a Markov property described by an undirected graph. MRF is similar to a Bayesian network in its representation of dependencies. In case of stereo matching, MRF considers not only intensity differences between stereo pairs but also the relation between neighboring disparity values. Since MRF-based methods are formulated in an energy-minimization frame work, they face with the global optimization problem.

In the MRF model, a current disparity value at a certain position affects its neighbors. Therefore, the conventional winner-takes-all algorithm is not able to solve it. In addition, it is physically impossible to calculate the exact solution from whole candidates. When the size of an input image is 100 x 100 and the possible disparity range is 10, the number of candidates is 10^{10000}. It is a tremendous number.

In order to simply approximate the solution, various algorithms such as the belief propagation (BP) [3], graph cut [4], and dynamic programming [5] have been proposed. Among them, the BP shows good results and is widely used. It has been found to yield accurate results, but despite recent advances is often too slow for practical use.

The execution time leaves still room for improvement. Several researches focus on fast computation. Felzenszwalb *et. al.* presented algorithmic techniques that substantially improve the running time of the BP approach [6]. They use a hierarchical structure to reduce the number of message passing iterations to a small constant rather than being proportional to the diameter of the image grid graph.

Petersen *et. al.* proposed a fast generalized BP algorithm for maximum a posterior estimation on 2D and 3D grid-like MRF [7]. They developed a caching method that significantly reduces the number of multiplications during generalized BP inference, and introduced a speed-up for computing the MAP estimate of general BP cluster messages by presorting its factors and limiting the number of possible combinations.

M. P. Kumar *et. al.* proposed methods for reducing both run time and storage needed by BP for a large class of pairwise potentials of the MRF [8]. Further, they show how

the problem of sub-graph matching can be formulated using this class of MRFs.

Recently, researchers have presents the methods using additional hardware such as a graphics processing unit (GPU) and field-programmable gate array (FPGA) architecture for further improvement [9][10].

In this paper, we implement the hierarchical BP algorithm on a GPU using compute unified device architecture (CUDA). It is suitable for practical use. The initial message memory in coarse level is uploaded to a global memory in a GPU, and the messages for fine level are calculated in parallel. The relation between block size and processing time is analyzed.

II. Parallel Programming and Global Optimization

We start by briefly reviewing the CUDA and the hierarchical BP approach for performing inference on Markov random field.

A. CUDA

Recently, the programmable graphics processor unit has evolved into an absolute computing workhorse. With, The GPUs offer incredible resources for both graphics and non-graphics processing, because they have multiple cores driven by very high memory bandwidth [11].

The GPU is compute-intensive, highly parallel computation for graphics rendering, and it is designed such that more transistors are devoted to data processing rather than data caching and flow control. Therefore, the GPU is especially suited to address problems. The problem is expressed as data-parallel computations with high arithmetic intensity. Data-parallel processing maps data elements to parallel processing threads. In order to speed up, many applications that process large data sets use a data-parallel programming model.

In 3D rendering process, many pixels and vertices are mapped to parallel threads. Similarly, image and media processing applications such as post-processing of rendered images, video encoding and decoding, image scaling, stereo vision, and pattern recognition can map image blocks and pixels to parallel processing threads. Therefore, various applications of image processing are suitable for parallel processing. Other algorithms outside the field of image rendering and processing are also accelerated by data-parallel processing.

Various programming libraries have been developed for data-parallel processing. Among them, CUDA is a very popular and powerful library. CUDA is NVIDIA's parallel computing architecture that enables dramatic increases in computing performance by harnessing the power of the GPU.

CUDA gives developers access to the virtual instruction set and memory of the parallel computational elements in CUDA GPUs. Using CUDA, the latest NVIDIA GPUs become accessible for computation like CPUs. Unlike CPUs however, GPUs have a parallel throughput architecture that emphasizes executing many concurrent threads slowly, rather than executing a single thread very quickly. This approach of solving general purpose problems on GPUs is known as GPGPU. Figure 2 shows the different structures between CPU and GPU.

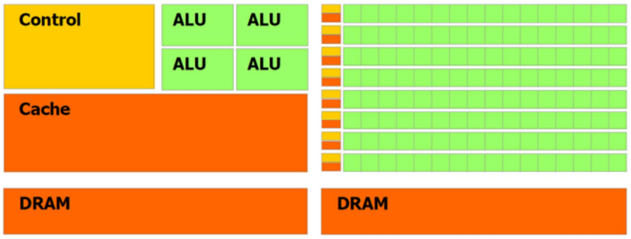


Figure 2. Structures of CPU and GPU

B. Hierarchical Belief Propagation

The general framework of the hierarchical BP implemented in this paper is defined as follows [6]. The loopy BP has high memory requirements, storing multiple distributions at each pixel. In [6], it is presented that a multi-grid approach for performing BP in a coarse-to-fine manner. In this approach the number of message passing iterations can be a small constant, because long range interactions are captured by short paths in coarse grid graphs.

Let P be the set of pixels in an image. The labels correspond to disparity values that we want to estimate at each pixel. A labeling f assigns a label f_p to each pixel p. It is assumed that the disparity values should vary smoothly almost everywhere but may change dramatically at some places such as object boundaries. The quality of a labeling is given by an energy function,

$$E(f) = \sum_{p,q \in N} V(f_p, f_q) + \sum_{p \in P} D_p(f_p), \qquad (1)$$

where N are the edges in the four-connected image grid graph. $V(f_p, f_q)$ is the cost of assigning labels f_p and f_q to two neighboring pixels, and is normally referred to as the discontinuity cost. $D_p(f_p)$ is the cost of assigning label f_p to pixel p, which is referred to as the data cost. Finding a labeling with minimum energy corresponds to the MAP estimation problem for an appropriately defined MRF.

The hierarchical BP reduces the time required to compute a single message update from $O(k^2)$ to $O(k)$. Each message is represented by

$$m_{pq}^t(f_q) = \min_{f_p}(V(f_p, f_q) + h(f_p)) \qquad (2)$$

The standard way of computing the messages is to explicitly minimize over f_p for each choice of f_q. This takes $O(k^2)$ time, where k is the number of labels. However, in stereo matching the cost $V(f_p, f_q)$ is generally based on some measure of the difference between the labels f_p and f_q rather than on the particular pair of labels. In such cases the messages can often be computed in $O(k)$ time. BP can be performed more efficiently for a bipartite graph with node A and B as

$$m_{pq}^t = \begin{cases} m_{pq}^t & if\ p \in A\ (if\ p \in B) \\ m_{pq}^{t-1} & otherwise \end{cases}. \qquad (3)$$

Figure 3 shows the hierarchical BP structure, and the hierarchical structure reduces the number of message passing iterations.

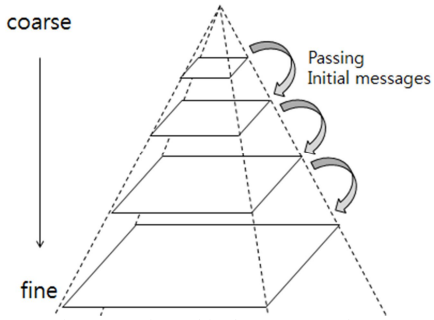


Figure 3. Hierarchical structures of BP

III. Hierarchical Belief Propagation on GPU

We do not implement the part calculating data terms because it takes a short time compared with the hierarchical BP part. The computed data terms on the CPU are uploaded to the global memory of the GPU, and the initial messages are also uploaded. With the data, we calculate and update the further messages on the GPU in parallel.

In our implementation, each scale level has the similar process. For each scale, eight textures are used to store the smoothness term. The four textures are from the previous iteration, and other four textures are from current iteration.

To reduce the bottleneck effects of conditional parts, we replace them with simple calculations. In addition, the CPU and GPU do not exchange unnecessary data, because the speed of data exchange between *Host* and *Device* units is very slow. Figure 4 shows the strucure of kernels, blocks and threads. The arrangement of threads in blocks and blocks into grids of blocks is up to the developer. We divide an input image into rectangular blocks, and find the optimal block size for hirarchical BP.

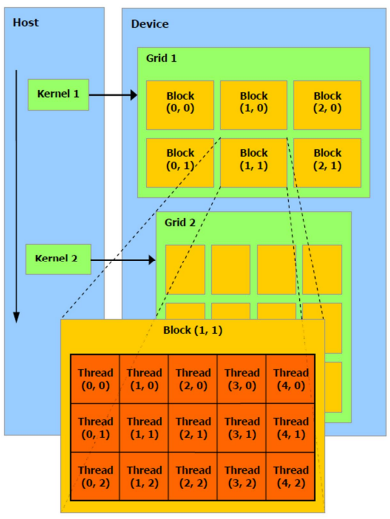


Figure 4. Structures of CPU and GPU

IV. Experimental Results

The hierarchical BP algorithm implemented by CUDA is based on the algorithm introduced in [6]. At first, we measured processing time with various block sizes. Table I shows the result with three test images. The test image is Newspaper, and it was down-sampled for different experimental conditions.

TABLE I
PROCESSING TIME ACCORDING TO BLOCK SIZES

Image size	Block size	Time(s)
320 x 240 (disparity range: 30)	1 x 1	0.99
	2 x 2	0.44
	3 x 3	0.35
	4 x 4	**0.32**
	5 x 5	0.38
640 x 480 (disparity range: 40)	1 x 1	4.77
	2 x 2	1.59
	3 x 3	1.02
	4 x 4	**0.91**
	5 x 5	0.94
1024 x 768 (disparity range: 50)	1 x 1	9.42
	2 x 2	4.84
	3 x 3	4.02
	4 x 4	**3.26**
	5 x 5	3.35

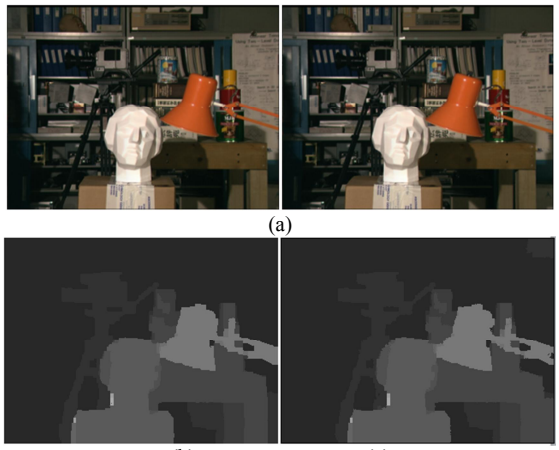
The block size which shows the best performance is 4 x 4, and the optimal block size does not depend on the input image size. The larger block sizes decrease the performance or stop operating. We set the block size with 4 x 4, and had the next experiments.

Figure 5(a), Fig. 6(a), and Fig. 7(a) show the input stereo pairs: the first is the 320 x 240 Tsukuba image, the second is the 640 x 480 Cone, and the third is the 1024 x 768 Newspaper. Tsukuba and Cone images are downloaded from the Middlebury site, and they are resized for quantitative experiments. We use five hierarchical levels and five iterations per level.

TABLE II
COMPARISON OF PROCESSING TIME

Image size	Methods	Processing Time (s)
Tsukuba 320 x 240 (disparity range: 20)	Loopy BP	7.02
	Hierarchical BP on CPU	0.81
	Hierarchical BP on GPU	**0.34**
Cone 640 x 480 (disparity range: 80)	Loopy BP	295.02
	Hierarchical BP on CPU	10.01
	Hierarchical BP on GPU	**2.29**
Newspaper 1024 x 768 (disparity range: 60)	Loopy BP	475.64
	Hierarchical BP on CPU	20.37
	Hierarchical BP on GPU	**5.17**

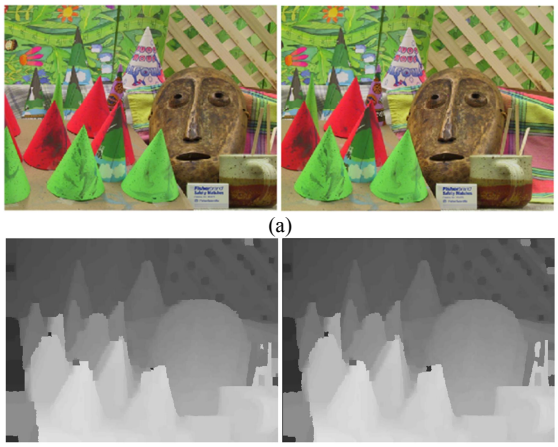
As shown in Table II, our implementation provides the fastest processing time compared with the loop BP and the hierarchical BP algorithms on CPU. For the experiment, we use the desktop computer which is equipped with an Intel Core 2 Quad CPU running at 2.4 GHz and nVidia Geforce GTX 470 GPU. The GPU has 488 CUDA cores and 1280 MB standard memory.



(a)

(b) (c)

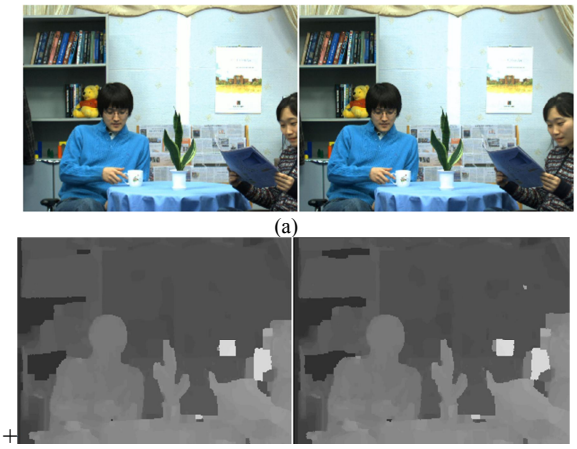
Figure 5. Tsukuba: (a) input and disparity maps from hierarchical BP on (b) CPU and (c) GPU



(a)

(b) (c)

Figure 6. Cone: (a) input and disparity maps from hierarchical BP on (b) CPU and (c) GPU



(a)

(b) (c)

Figure 7. Newspaper: (a) input and disparity maps from hierarchical BP on (b) CPU and (c) GPU

The ratio of speed rapidly increases when the image size and disparity range are large. In case of the small image, the data exchanging time between *Host* and *Device* units occupies the relatively large portion. The data exchanging time is a weak point of the parallel processing on GPU.

Figure 5(b) and Fig. 5(c) are the disparity maps calculated by hierarchical BP on CPU and GPU, respectively. Despite the processing time difference, our implementation provides the disparity maps of similar quality. Figure 6 and Fig. 7 are the additional results of Cone and Newspaper images.

V. Conclusion

The BP algorithm effectively infers global solutions, but it takes a very long time to calculate messages. In this paper, we implemented the hierarchical BP algorithm on GPU using CUDA. We upload data terms and initial messages to the global memory of the GPU, and compute whole messages in parallel. To reduce processing time, we minimize unnecessary date exchanges between *Host* and *Device* units. We analyzed the processing time according to grid and block sizes. Experimental results show that our implementation is faster than the loopy BP algorithm and the hierarchical BP algorithm on CPU.

Acknowledgment

This research is supported by Ministry of Culture, Sports and Tourism (MCST) and Korea Creative Content Agency (KOCCA) in the Culture Technology (CT) Research & Development Program 2010.

References

[1] E. Lee and Y. Ho, "Generation of High-quality Depth Maps using Hybrid Camera System for 3-D Video," Journal of Visual Communication and Image Representation, vol. 22, issue 1, pp. 73-84, Jan. 2010.

[2] D. Scharstein and R. Szeliski, "A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms," International Journal of Computer Vision, vol. 47, no. 1-3, pp. 7-42, April 2002.

[3] J. Sun, N. Zheng, and H. Shum, "Stereo Matching Using Belief Propagation," IEEE Transactions on pattern analysis and machine intelligence, vol. 25, no. 7, pp. 787-800, July 2003.

[4] Y. Boykov, O. Veksler, and R. Zabih, "Fast Approximate Energy Minimization via Graph Cuts," IEEE Transactions on pattern analysis and machine intelligence, vol. 23, no. 11, pp. 1222-1239, Nov. 2001.

[5] O. Velsler, "Stereo Correspondence by Dynamic Programming on a Tree," Computer Vision and Pattern Recognition, vol. 2, pp. 384-390. June 2005.

[6] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient Belief Propagation for Early Vision," International Journal of Computer Vision, vol. 70, no. 1, pp. 41-54. Oct. 2006.

[7] K. Petersen, J. Fehr, and H. Burkhardt, "Fast Generalized Belief Propagation for MAP Estimation on 2D and 3D Grid-Like Markov Random Fileds," Deutsche-Arbeitsgemeinschaft-fur-Mustererkennung Symposium on Pattern Recognition, pp. 10-13, June 2008.

[8] M. P. Kumar and P. Torr, "Fast Memory-Efficient Generalized Belief Propagation," European Conference on Computer Vision, pp. 451-463, May 2006.

[9] J. Pérez, P. Sánchez, and M. Martinez, "High memory throughput FPGA architecture for High-Definition Belief-Propagation Stereo Matching," International Conference on Signals, Circuits and Systems, pp. 1-6, Nov. 2009.

[10] Q. Yang, L. Wang, R. Yang, S. Wang, M. Liao, and D. Nister, "Real-time Global Stereo Matching Using Hierarchical Belief Propagation," The British Machine Vision Conference, pp.1-8, Sept. 2006.

[11] CUDA Reference Manual, Nvidia, July 2009.