Memory-Efficient Belief Propagation in Stereo Matching on GPU

Young-kyu Choi, Williem, and In Kyu Park
Inha University, Incheon 402-751, Korea
E-mail: {ykchoi@cs.ucla.edu, williem_060689@hotmail.com, pik@inha.ac.kr}

***Abstract*—Belief propagation (BP) is a commonly used global energy minimization algorithm for solving stereo matching problem in 3D reconstruction. However, it requires large memory bandwidth and data size. In this paper, we propose a novel memory-efficient algorithm of BP in stereo matching on the Graphics Processing Units (GPU). The data size and transfer bandwidth are significantly reduced by storing only a part of the whole message. In order to maintain the accuracy of the matching result, the local messages are reconstructed using shared memory available in GPU. Experimental result shows that there is almost an order of reduction in the global memory consumption, and 21 to 46% saving in memory bandwidth when compared to the conventional algorithm. The implementation result on a recent GPU shows that we can obtain 22.8 times speedup in execution time compared to the execution on CPU.**

I. Introduction

Stereo matching is one of the most fundamental technique that can be used for 3-dimensional reconstruction. The problem is first formulated as a label assignment problem in a probabilistic graphical model such as Markov random field (MRF). Next, it is solved by well-established global energy minimization algorithms such as belief propagation (BP) or graph cuts (GC).

However, one of the bottleneck of such global algorithms is that they require large memory bandwidth and space. This problem becomes more severe in BP algorithm. Although BP has advantage in terms of its intrinsic parallel characteristic, it requires large memory space that increases linearly with both the number of pixels (N) and the number of labels (L). This is much larger than GC, which only requires memory space linear to N.

Thus, our aim is to devise an efficient BP algorithm which is suitable for memory-efficient MRF-based energy minimization. We concentrate on two aspects: reducing data size and transfer bandwidth. There are two common approaches to solve this problem, namely reducing number of labels (*i.e.* reduction in disparity range) [7][8] and reducing number of pixels (*i.e.* reduction in spatial domain) [3]. The proposed method follows the second type of approach. Note that these techniques are not exclusive of each other, but can be combined to obtain better result.

In this paper, we propose a novel memory-efficient algorithm for BP. The data size and transfer bandwidth is significantly reduced by storing only a part of the whole message. This approach, however, reduces the accuracy of the estimated disparity map. In order to maintain the accuracy, the local messages are reconstructed by taking advantage of the shared memory available in Graphic Processing Units (GPU). The experimental result shows that the proposed method requires less memory space and bandwidth than conventional algorithms. It also shows that we can obtain high speedup when the proposed method is implemented on GPU.

This paper is organized as follows. We briefly describe how belief propagation is used to perform stereo matching in Section II. In Section III, the proposed method is explained in detail. The experimental results are shown in Section IV. Finally, we give a conclusive remark in Section V.

II. Belief Propagation in Stereo Matching

Stereo matching is a process of finding the distance to objects with two cameras. The distance is measured by finding the correspondence between pixels in the left and right images. This matching problem can be solved by minimizing the energy function E, which models the cost of assigning disparity label to each pixel. E is composed of data matching cost energy E_d and smoothness energy E_s:

$$E = E_d + E_s = \sum_p d_p(l_p) + \lambda \sum_{(p,q)} V_{p,q}(l_p, l_q) \qquad (1)$$

where $d_p(l_p)$ is the matching cost of assigning label l_p to pixel p. $V_{p,q}(l_p, l_q)$ is the smoothness cost of assigning labels l_p and l_q to neighboring pixels p and q, and λ is the weight of the smoothness cost. Note that we use the truncated linear function to model the smoothness cost.

E can be minimized by loopy BP algorithm [5]. BP operates by passing messages to the neighbors. A new message is constructed by the following equation:

$$m^t_{p\to q}(l_q) = min_{l_p}[d_p(l_p) + \lambda V_{p,q}(l_p, l_q) + \sum_o m^{t-1}_{o\to p}(l_p)] \qquad (2)$$

where t and $t-1$ denotes the current and previous iteration, and $m^{t-1}_{o\to p}(l_p)$ refer to the messages from neighboring pixels of p except the pixel q. This message passing process is repeated until the algorithm converges.

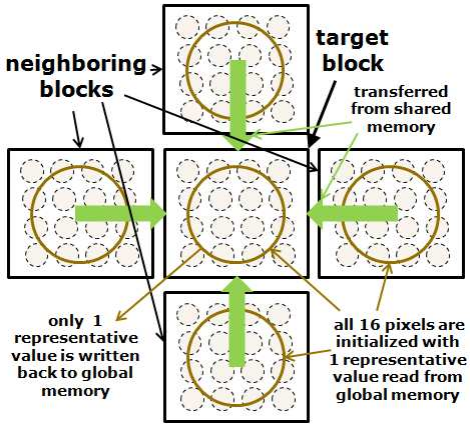


Fig. 1. Data transfer in the proposed algorithm.

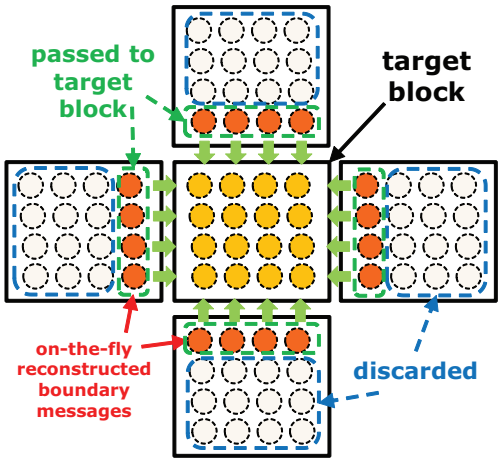


Fig. 2. On-the-fly boundary message reconstruction.

III. On-the-fly Reconstruction of Local Messages

A. Principles

Due to the ever-increasing difference between slow memory and fast compute devices, many modern computing systems are heavily utilizing local memory. Examples of such system includes shared memory in GPUs, block RAMs in FPGAs, and scratchpad memory in embedded systems. In this paper, we take advantage of such architecture to reduce memory bandwidth and the global memory size.

The intuition of the proposed method comes from the observation that most messages in the neighboring pixels are highly correlated. This suggests that we could use only one message value to represent the messages in the neighborhood. However, performing BP based on such naive approach will inevitably reduce the quality of the estimated disparity map. Thus, we propose a new method of reconstructing the local messages using shared memory in GPU.

The local messages are reconstructed with the block's representative message and each pixel's matching cost. For example, if the image is divided into 4×4 pixel blocks, only 1 representative message is read or written to the global memory. However, the matching costs for all 16 pixels are used to reconstruct the local messages. For the representative message, we simply use averaged value of the 16 messages. Note that the matching cost is computed on-the-fly using left/right image intensity values, as suggested in [3].

In order to reconstruct messages, first, all pixels within the block are initialized with the representative value. Next, we iteratively perform message passing within the block until convergence. However, simply performing message passing within the block will not work correctly, because there is no information about the messages from outside the block. Thus, we also have to fetch messages from the outside the block. As a result, a total of 5 representative value (up, down, left, right, center), along with the intensity of each pixel, is fetched from the global memory. This process is depicted in Fig. 1.

B. Boundary Message Reconstruction

The problem with the proposed method explained in the previous section is that the representative message from neighboring block only has low-resolution information. It does not have detailed information of each pixel.

Thus, we propose on-the-fly reconstruction of the neighboring blocks' pixel-wise messages, in addition to the reconstruction of messages of the current block. The neighboring blocks' message reconstruction process is similar to that of the current block - the message passing is performed until convergence. After convergence, only the message to the current block is saved - the rest of the messages is discarded, as shown in Fig. 2.

From the perspective of [3], the proposed algorithm can be seen as on-the-fly boundary message reconstruction. [3] proposes *storing* all boundary message, whereas the proposed algorithm saves memory space by only storing representative message and *reconstructing* all boundary messages. Due to the reconstructed boundary messages, the proposed algorithm still maintains comparable energy minimization performance. The proposed algorithm requires less memory space and bandwidth since only a singular value is stored among all pixels of this block.

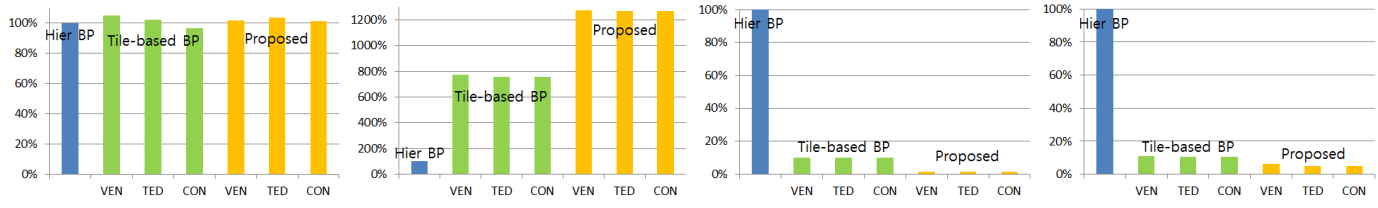
Note that due to the additional message passing in the neighboring blocks, the computation demand increases compared to original BP. However, this overhead is alleviated by reduced data bandwidth and abundant compute devices in GPUs.

C. Implementation Details

For implementation, we have to decide 3 types of iteration number: the number of iteration performed to pass messages among the blocks (outer iter num), the number of iteration performed inside the block (inner iter num), and the number of iteration performed to construct boundary message inside the block (boundary iter num). The final energy level decreases as these iteration number increases, at the cost of additional computation. These parameters were decided experimentally

TABLE I
EXPERIMENTAL RESULT ON TSUKUBA IMAGE

	Block Size	Hier. level	Outer iter	Inner iter	Boundary iter	Bad Pixel (%)	Final Minimized Energy	Num of Message Comp. (M)	Consumed Local Memory (KB)	Consumed Global Memory (MB)	Local↔Global Data TRX(MB)
Hier BP-M	-	6	5		-	1.75	303,173	2.9	-	34	765
Tile BP	4×4	-	5	5	-	3.45	293,327	22	5.4	6.8	160
	8×8	-	5	5	-	3.08	296,479	22	21	3.4	84
	16×16	-	5	10	-	2.77	296,973	44	83	1.7	46
Proposed	4×4	4	5	5	3	1.98	309,427	37	7.4	1.7	127
	8×8	3	5	5	3	2.33	298,773	37	26	0.42	48
	16×16	2	5	10	5	3.13	290,515	64	97	0.11	25



(a) (b) (c) (d)

Fig. 3. Comparison between hierarchical BP-M, tile-based BP, and the proposed method on Tsukuba (384×288), Venus (434×383), Teddy (450×375), Cones images (450×375). The block size is 8×8. (a) Minimized energy. (b) Number of message computation. (c) Consumed global memory. (d) Global↔local data transaction.

by increasing its value until the final energy converged to a minimum point.

For fast convergence, we apply hierarchical method [2] before performing the proposed algorithm. For example, if the target block size is 8×8, the hierarchical BP from level 6 (64×64 block) to level 3 (8×8 block) would be performed first. Then, the proposed algorithm would be applied. This technique improves the minimization performance at the cost of small increase in execution time.

We adopt asynchronous update scheme proposed in [6] ('BP-M') to accelerate convergence. We also use fast message update technique in [2] to reduce the amount of computation.

IV. EXPERIMENTAL RESULTS

A. Experimental Setup

For experiment, we use Tsukuba, Venus, Cones, Teddy stereo test images from Middlebury website [4]. The input images were smoothed with Gaussian filter $\sigma = 0.7$, and we use smoothness weight of $\lambda = 8, 4, 20, 15$ for Tsukuba, Venus, Teddy, Cones images, respectively. The test result for Tsukuba image is shown in detail in Table I. The result for the rest of the images is shown in Fig. 3.

Table I and Fig. 3 show the result for energy minimization performance, memory resource consumption, and computational complexity. For comparison purpose, they also show the result for our implementation of hierarchical BP-M and tile-based BP [3]. The values for tile-based BP and the proposed method in Fig. 3 are normalized to that of hierarchical BP-M.

Note that the number of message passing computation and global memory transfer was obtained by reading from a counter that was triggered whenever such event has occurred. Also, for fair comparison, we have matched the iteration number of tile-based BP and the proposed method.

B. Performance

The performance of the compared algorithms is shown as the final minimized energy level in Table I and Fig. 3 (a). It can also be judged from the percentage of non-occluded pixels that has more than 1 label difference from the true disparity map ('bad-pixel'). The result shows that the proposed method has comparable performance to the original hierarchical BP-M and the tile-based BP. As an example, the disparity maps for Tsukuba and Venus images are shown in Fig. 4.

C. Memory Resource Consumption

If the total number of pixels in the entire image is N and the number of pixels in the block is M, the total global memory size requirement of the proposed algorithm is N/M. This is large improvement over the hierarchical BP-M (N) and the tile-based BP ($4N/M^{\frac{1}{2}}$). As mentioned in Section III, this improvement is from the fact that only signle pixel out of M pixels requires data storage space in the global memory. Table I and Fig. 3 (c) confirm that there is almost an order of magnitude reduction. The saving becomes larger as the block size increases. As a result, the proposed method allows higher resolution images to fit into systems with smaller global memory space.

Since only single representative value is read from or written to global memory, the proposed algorithm also saves local to global data transfer bandwidth. As observed in Table I and

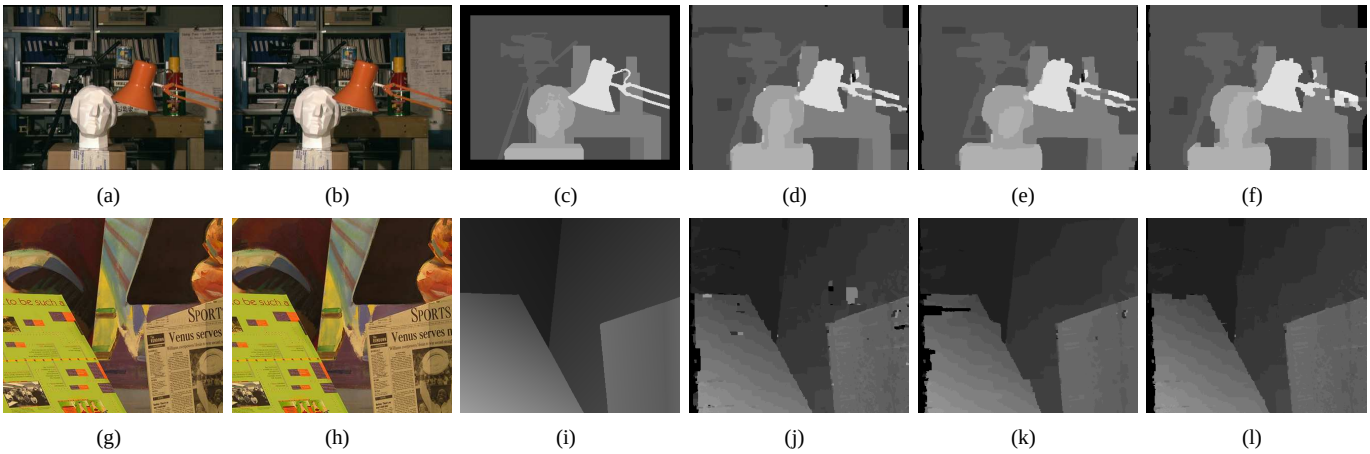


Fig. 4. Depth map obtained for Tsukuba and Venus testset. (a)(g) Left image. (b)(h) Right image. (c)(i) Ground truth. (d)(j) Hierarchical BP-M. (e)(k) Tile-based BP. (f)(l) Proposed algorithm.

Fig. 3 (d), the proposed method requires 21 to 46% less bandwidth than the tile-based BP, and much less than the hierarchical BP-M. As a result, the proposed method will cause less memory contention problem and leads to faster execution time in systems with memory bottleneck problem.

D. Computational Complexity

It is known that the total amount of computation is almost proportional to the number of message passing computation [1]. Thus, the number of message passing computation is provided in Table I and Fig. 3 (b) to infer the computational complexity. They show that the amount of computation is increased by about 45 to 68% compared to the tile-based BP. Such overhead, which was caused by the on-the-fly reconstruction of the boundary messages, is the main limitation of the proposed algorithm.

E. GPU Implementation

We have implemented the proposed algorithm on NVIDIA GTX580 GPU running at 3.91GHz with 3GB global memory. It has 512 CUDA cores in 16 streaming multiprocessors and 48KB shared memory in each multiprocessor. For comparison with the baseline algorithm, we implemented hierarchical BP on Intel CPU i7 2600. Note that the small shared memory on GPU has limited the implementation choice to using only 8×8 block.

In order to exploit massive parallelism in GPU architecture, we have used synchronous update scheme in [2]. As a result, all pixels can perform message passing independently. For higher parallelism, the message update to 4 neighboring pixels is also parallelized, and each update is performed by independent threads.

Experiment shows that the execution time for Tsukuba testset is 4.38 and 0.192 seconds on CPU and GPU, respectively. This corresponds to 22.8 times speedup. The main cause is due to the GPU's massive parallel compute units and global memory bandwidth reduction by the proposed algorithm. As the future GPUs incorporate more compute units and memory bottleneck problem becomes more severe, we expect further savings with the proposed algorithm.

V. Conclusion

In this paper, we have described a new memory-efficient algorithm of BP in stereo matching. We have proposed on-the-fly local message reconstruction method using matching cost and reconstructed boundary message. The experimental result on Middlebury test images showed that, at the cost of about 45 to 68% increase in the computation, almost an order of magnitude savings in global memory space and 21 to 46% savings in memory bandwidth was obtained compared to the conventional algorithm. GPU implementation of the proposed algorithm shows 22.8 times speedup in execution time.

References

[1] Y. Choi, "CUDA implementation of belief propagation for stereo vision," in *Proc. IEEE Int. Conf. Intelligent Transportation Systems*, pp. 1402–1407, Sept. 2010.

[2] P. Felzenszwalb and D. Huttenlocher, "Efficient belief propagation for early vision," *Int. J. Computer Vision*, vol. 70, no. 1, pp. 41–54, Oct. 2006.

[3] C.-K. Liang, C.-C. Cheng, Y.-C. Lai, L.-G. Chen, and H. H. Chen, "Hardware-efficient belief propagation," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 21, no. 5, pp. 525–537, May 2011.

[4] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *Int. J. Computer Vision*, vol. 47, no. 1–3, pp. 7–42, Apr. 2004.

[5] J. Sun, H. Shum, and N. Zheng, "Stereo matching using belief propagation," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 25, no. 7, pp. 787–800, July 2003.

[6] M. Tappen and W. Freeman, "Comparison of graph cuts with belief propagation for stereo, using identical MRF parameters," in *Proc. IEEE Int. Conf. Computer Vision*, vol. 2, pp. 900–906, Oct. 2003.

[7] Q. Yang, L. Wang, and N. Ahuja, "A constant-space belief propagation algorithm for stereo matching," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 1458–1465, June 2010.

[8] T. Yu, R.-S. Lin, B. Super, and B. Tang, "Efficient message representations for belief propagation," in *Proc. IEEE Int. Conf. Computer Vision*, pp. 1–8, Oct. 2007.