Fast and Adaptive BP-based Multi-core Implementation for Stereo Matching

Armin Ahmadzadeh, Hatef Madani, Kianoush Jafari, Farzad Salimi Jazi, Shervin Daneshpajouh, Saeid Gorgin

School of Computer Science,
Institute for Research in Fundamental Sciences (IPM),
Tehran, Iran
Email: {gorgin,daneshpajouh}@ipm.ir

***Abstract*—The stereo matching problem has been under attention of numerous researchers for many years, due to the wide range of applications in computer vision. The MEMOCODE 2013 design contest was aimed to develop a very fast and efficient stereo matching method to infer the depth information for each pixel of a pair stereo image via Belief Propagation on Markov Random Field graph. We first explored different platforms, *i.e.* FPGA, GPU, multi-core CPU, to find the best one for solving the contest problem. Based on our investigation, we select multi-core system and design and implement a solution on this platform. We use a series of techniques and optimization methods to improve the running time of the stereo matching problem while preserving the error compared to the given reference solution. Our method outperform other team's solutions in both classes: absolute-performance and cost-adjusted-performance.**

I. Introduction

Stereo matching is one of the important components of the computer vision. It is a label-assignment problem to infer the depth information of 2D images by relying on parallax effect phenomena. It simply expresses that closer objects move quicker than those further away; that is the pixels in the closer objects have greater amount of disparity than the other one [1]. In this problem, we are given a pair of left and right images which are taken from slightly different horizontal views. The goal of stereo matching is to label the depth information by processing these two images. The output of the algorithm is a grayscale image in which every pixels with the same depth have the same color; the closest ones are white and the furthest are black.

It is proven that the Label-assignment problem is NP-hard [5]. There exist some approximation methods that model this problem using Markov Random Field (MRF). The MRF graph can be solved by global energy minimization algorithms such as Belief Propagation (BP) or graph-cuts. Although these algorithms grantee the approximation factor, their running time and memory requirements are too high for real world applications. The subject of MEMOCODE 2013 hardware/software co-design contest is stereo matching using belief propagation (BP) algorithm [1]. Belief Propagation is a message passing algorithm that was originally used for acyclic graphs [2], which returns the exact result of energy minimization. It can also be used for loopy graphs, known as Loopy Belief Propagation (LBP), and it has been successfully applied to many problem domains such as early vision [3], [4], [5].

In each iteration of BP algorithm, it propagate messages between adjacent nodes of MRF graph using an updating schema. There are two main variant of update schema: BP-M and BP-S [4]. In BP-M each node updates its adjacent nodes with highest belief in four different directions (left, right, up, and down). This update formula is shown in the following equation.

$$m^t_{p\to q}(f_q) = \min_{f_p}(V(f_p - f_q) + D_p(f_p) + \sum_{s\epsilon N(p)\setminus q} m^{t-1}_{s\to p}(f_p))$$

The other updating schema is BP-S which is derived form the TRWS [6]. The most important difference between BP-M and BP-S is their updating schedules for passing messages on grid. In BP-S messages are passed in forward and backward manner. In forward pass, each message can move in right and down till reaches to the last nodes. In backward pass, each message can move to the left and up till it reaches to the first node. See [4], [7], [8] for more detailed description of LBPs. Also, the BP-S schema is less accurate than BP-M [4], [9] and it is absolutely sequential that make it difficult for hardware parallelism.

In this contest, we chose multi-core CPU over GPU and FPGA. The data transfer between CPU and GPU is too high compared to the total running time of the algorithm on a multi-core system. Note that there is also another bandwidth bottleneck for accessing data on GPU's global memory by its cores. Also, for FPGA, not only it has data transfer limitations, but also, it is hard to place the whole design on the FPGA hardware.

The rest of paper is organized as follows: Section II presents a detailed description of the contest problem. We introduce our method which is the all-around winner of this year design contest in two sections: Section III shows optimization techniques and Section IV presents the implementation method. In Section V, we review the detailed information of our hardware platforms and present the experimental results for both benchmark images: Tsukuba and Barrels. Finally, we conclude in Section VI.

978-1-4799-0905-6/13/$31.00 ©2013 IEEE

II. Problem Description

As mentioned in the introduction section, this year's contest is about stereo matching problem with the aim of developing a fast and efficient BP-based optimization algorithm on MRF graph within the duration of one month. The most time-consuming part of the MRF-based solutions is its optimization section. The MRF graph is a grid of pixel-pair data as nodes, and consist of two parts: (1) Data part, (2) Messages part. Data part is initiated via input file that contains disparity values of each pixel. Messages part consists of message values of each node (Initially, these values are set to be zero). The goal is to output the disparity map, that is to identify the depth information by assigning a label to each node. The accuracy of the algorithm is defined as the number of mismatched labels between output of the algorithm and the provided reference code. The proposed algorithm should have the same or better accuracy than the standard BP on the reference data sets.

The BP algorithm needs some iterations till it converge. As an example, in the case of the Tsukuba dataset, it takes 40 iteration to converge. The time complexity of BP-M message construction is $O(k^2)$ where k is the number of labels. So, the total running time of the optimization algorithm is $O(t.n.k^2)$ where t is the number of the iterations and n is the number of nodes. Although BP-M is a popular method due to its simplicity and regularity, it has its own weaknesses. The amount of memory used by BP-M is at least 5-times larger than the input data. This issue not only makes the cache utilization difficult, but also causes some challenges to apply BP-M method to devices with limited memory.

III. Optimization techniques

In this section, we present a set of optimization techniques employed or tailored for our problem in order to reduce the running time (CPU wall time) of the naïve BP algorithm.

A. Min-convolution

The time complexity of traditional BP algorithm is $O(k^2)$. We employ the min-convolution algorithm presented in [5] to reduce the complexity down to $O(k)$. In this method, message labels are computed in two linear passes; forward pass and backward pass.

The forward pass is:

$$\begin{aligned}&\text{for } f_q \text{ from } 1 \text{ to } k-1,\\ &\quad m(f_q) \leftarrow \min(m(f_q), m(f_1-1)+c),\end{aligned}$$

and backward path is:

$$\begin{aligned}&\text{for } f_q \text{ from } k-2 \text{ downto } 0\\ &\quad m(f_q) \leftarrow \min(m(f_q), m(f_1+1)+c)\end{aligned}$$

For detailed description of this method see [5].

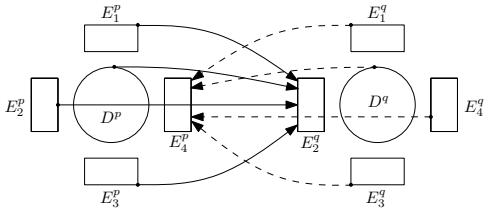


Fig. 1: Flow of bi-direct message passing. Circles represent two neighboring nodes of MRF grid (p,q). The message are passed to edges E_i of each node which are represented by rectangles. D demonstrates data part. The arrows represent the flow of bi-direct message passing, where dashed and solid ones represent the right-to-left and left-to-right, respectively.

B. Hierarchy of vertices

In large images, LBP needs large number of iterations to stabilize the messages between distant nodes. The number of iterations can be reduced using Hierarchical BP method of [5]. In fact, in this method the length of message passing between two distant nodes are reduced. Therefore, we employ [5] method and tailor it by reducing hierarchies into only two layers: coarse and fine. Fine-layer consists of MRF data and coarse-layer consists of MRF messages. Our experiments show that for the VGA image size having only two layers are sufficient. We compress the fine-layer data and send it to the coarse-layer. We perform the BP algorithm on the coarse-layer. Also, we need to transfer data of some specific nodes to the other one in the upper layer. Therefore, a specific mixing method is required. Following is our mixing equation.

$$D^{L_i} = \sum_{n=1}^{s^2} \frac{D_n^{L_{i-1}}}{df},$$

where $D_n^{L_i}$ represent data of n^{th} vertex number in layer L_i, s is the scaling factor of corresponding nodes in the fine layer and df is the division factor between 1 and 2 (*e.g.* $df = 1$ gives us sum of all nodes in the fine layer).

C. Bipartite method

We use bipartite method that splits main MRF nodes into two odd and even subsets [5], [12]. We observed that by skipping one subset in each iteration, the performance and accuracy improve. Therefore, at each iteration, we only transfer the messages of the nodes of only one subset.

D. Bidirectional message passing

In traditional BP method, the messages are passed in a certain direction and then vice versa (*e.g.* right-to-left, and then left-to-right). Clearly, there is no data dependency between these two message passing (See Fig. 1, in which forward and backward message passing have no data dependency). Therefore, message passing can be performed across both directions simultaneously.

IV. Implementation Method

Our implementation method consists of two parts: Computational and Memory optimizations. Following we explain these implementations.

A. Computational Optimization

Our computational optimization approaches consists of two basic parts: (1) We introduce an optimized solution for initialization of compact layer (coarse-layer) which consists of two arithmetic operations for each data element. We apply the operation mentioned in section III-B on the input data which transfer data to the coarser layer. We have parallelized relevant operations in order to speed-up the data generation for this layer. Our approach improves the running time by overlapping the memory latency and computational tasks. (2) We also optimize and parallelize message passing in BP algorithm. Different techniques can be considered to parallelize the BP message algorithm, depending on the architecture. Potentially, the algorithm can be parallelized in all four directions. Bi-direct method reduces these four steps (right, left, up, and down) to only two steps (row and column). Using this method, we reduce the number of steps of the algorithm from four to two, and consequently the computation time. Note that, the row and column steps cannot be run simultaneously, as they have data dependency to each other. We combine the observation that row (column) operations can be done simultaneously with the bi-direct method in order to fully utilize CPU cores. Note that, in this case the computational tasks become coarse grained, which is favourable for CPU cores. This also decrease the overhead of initializing a thread. We implemented these techniques using OpenMP API.

B. Memory Optimization

Memory bandwidth limitation still bounds the computational time. In order to improve the running time, we use two level of memory optimizations: (1) system level optimization for data initialization, and (2) Optimizing cache usage by employing bi-direct method.

We alter MRF data structure to a linear size data structure that can cover memory latencies with efficient cache usage. This structure stores input data in a separate vector, thus avoid loading irrelevant data into cache. We also used compiler directives to prefetch data from memory efficiently.

Using the bi-direct method, the cache hit ratio increases. In bi-direct method the left and right (top and down) messages pass simultaneously. Therefore, in half of updates, memory access has a minimal latency.

V. Our Results

In this section, we first overview our hardware platforms and then present our experimental results on these platforms.

A. Platforms Overview

Here is list of our multi-core platforms.

- Intel Core i5, M460 with two cores @ 2.57GHz.
- Intel Core i7, 960 with four cores @ 2.67GHz.

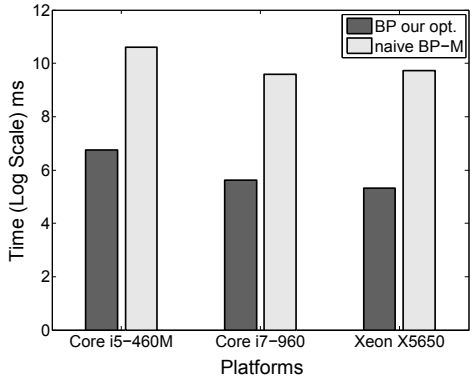


Fig. 4: Performance comparison of Config 40-1.

- Intel Xeon x5650 with six core @ 2.67GHz.

The hyper threading feature [13] is enabled in all of these platforms.

B. Experimental Results

We have experimented our implementation on aforementioned multi-core platforms on both Tsukuba and Barrels images. Table I summarize our results. Note that in this table scale factor is the scaling factor of corresponding nodes in the fine layer defined in Section III-B.

Our best time for Tsukuba is $0.6\mu s$ with 13695 mismatch (Table I-Config 4-8) which is less than 17743 mismatch of the reference code (See Fig. 2 for the output image comparison). Also, our best time for Barrels image is $204ms$ with 11833 (Table I-Config 40-1) which is less than 11870 mismatch of the reference code (See Fig. 3 for the output image comparison). Fig. 4 shows the performance comparsion for this config.

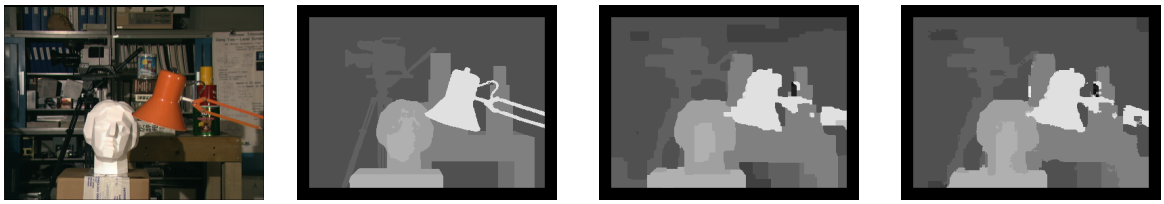
VI. Conclusion

In this paper, we have proposed a modified version of BP algorithm that improves the running time while the data accuracy is preserved. We employed a set of optimization techniques on multi-core platform in order to provide a faster and reliable solution. We evaluated our results on different platforms. Our experiments on Tsukuba test bench data shows that our method is more than four orders of magnitude faster than the reference solution.

Our solution is the all-around winner of this year's MEMOCODE design contest.

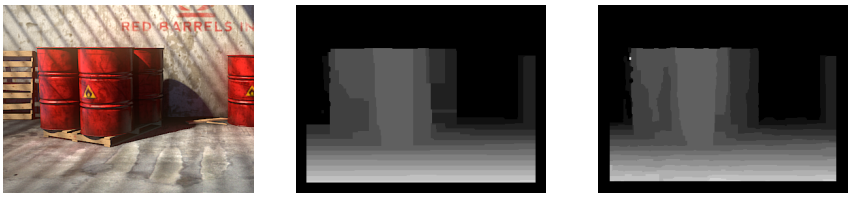
Acknowledgment

We are grateful to Prof. Hamid Sarbazi Azad, Head of school of computer science for his various support and useful guidance. We also would like to acknowledge Soroush Bateni, Ahmad Lashgar, Mohsen Mahmoudi Aznaveh, Alireza Majidi, Kamyar Mirzazad, Mehrshad Vosoughi, members of computer science school at Institute for Research in Fundamental Sciences (IPM) for their helpful discussions and comments.



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

Fig. 2: (a) Contest reference image. , (b) Tsukuba ground truth image. , (c) The reference code output with 17743 mismatches. , (d) Our method's output with 11367 mismatches.



(a) (b) (c)

Fig. 3: (a) Contest Barrels image., (b) The reference code output with 11870 mismatches. , (c) Our implementation's output with 11833 mismatches.

TABLE I: Comparison of implementations with different configuration on different multi-core platform.

Name	Iteration	Scale Factor	Mismatch (Tsukuba).	Mismatch (Barrels).	Time Req (Xeon X5650).	Time Req (Core i7-960).	Time Req (Core i5-460M).
Refrence Code	40	-	17743	11870	16.7 s	14.7 s	40.5 s
Config 4-8	4	8	13695	17084	0.6 ms	0.8 ms	2.9 ms
Config 4-4	4	4	15047	15348	1.1 ms	1.5 ms	5.9 ms
Config 40-4	40	4	11367	15328	6.3 ms	8.1 ms	40 ms
Config 40-1	40	1	12727	11833	204 ms	277 ms	850 ms
Config 14-1	14	1	17487	13318	73 ms	98 ms	300 ms
Config 20-1	20	1	16333	12570	103 ms	103 ms	426 ms
Config 30-1	30	1	14073	12048	155 ms	139 ms	635 ms

REFERENCES

[1] MEMOCODE 2013 Hardware/Software Co-design Contest: Stereo Matching. Available at: $http://memocode.irisa.fr/2013/$.

[2] J. Pearl, *probabilistic Reasoning In Intelligent Systems:Networks of plausible inference*, Morgan Kaufmann, (1988).

[3] B. Frey, D. Mackay, *A Revolution: Belief Propagation In Graphs With Cycles*, Advances in Neural Information Processing Systems, (1997).

[4] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, C. Rother, *A Comparative Study of Energy Minimization Methods For Markov Random Field with Smoothness-Based Priors*, IEEE Transaction on pattern analysis and machine intelligence, 30(6): 1068-1080 (2008).

[5] P. F. Felzenszwalb and D. P. Huttenlocher *Efficient Belief Propagation for Early Vision*, International Journal of Computer Vision 70(1): 41-54 (2006).

[6] M. Wainwright, T. Jaakkola, A. Willsky, *Map Estimation Via Agreement On Trees: Message-passing And Linear Programming*, IEEE Transactions on Information Theory, 51(11) 6697-3717 (2005)

[7] W. Freeman, E. Pasztor, O. Carmichael, *Learning Low-Level Vision*, International Journal Of Computer Vision, 40(1): 25-47 (2000).

[8] Loopy belief propagation, Markov Random Field, stereo vision, Available at: $http://nghiaho.com/?page\_id = 1366$.

[9] M. F. Tappen, W. T. Freeman, *Comparison of Graph Cuts with Belief Propagation for Stereo, Using Identical MRF Parameters*, Ninth IEEE International Conference on Computer Vision. 900-906 (2003).

[10] C. Liang, C. Cheng, Y. Lai, L. Chen, H. Chen, *Hardware-Efficient Belief Propagation*, IEEE Transaction On Circuits And Systems For Video Technology, 21(5): 525-537 (2011).

[11] Y. Tseng, N. Chang, T. Chang *Low Memory Cost Block-Based Belief Propagation for Stereo Correspondence*, IEEE International Conference on Multimedia and Expo. 1415-1418 (2007)

[12] A. Brunton, C. Shu and G. Roth, *Belief Propagation on the GPU for Stereo Vision*, In Proceedings of the 3rd Canadian Conference on Computer and Robot Vision (CRV'06), IEEE Computer Society, (2006).

[13] Intel Product Information, Available at: http://ark.intel.com.

[14] Middlebury benchmark, Available at: $http://vision.middlebury.edu/stereo/$.