Real-time Global Stereo Matching Using Hierarchical Belief Propagation

Qingxiong Yang[1] Liang Wang[1] Ruigang Yang[1] Shengnan Wang[2] Miao Liao[1] David Nistér[1]
[1]Center for Visualization and Virtual Environments, Department of Computer Science,
University of Kentucky, Lexington, KY, 40506, U.S.A.
[2]Department of Electronic Science and Technology,
University of Science & Technology of China, Hefei, China, 230026
qingxiong.yang@uky.edu
http://www.vis.uky.edu/~liiton

Abstract

In this paper, we present a belief propagation based global algorithm that generates high quality results while maintaining real-time performance. To our knowledge, it is the first BP based global method that runs at real-time speed. Our efficiency performance gains mainly from the parallelism of graphics hardware,which leads to a 45 times speedup compared to the CPU implementation. To qualify the accurancy of our approach, the experimental results are evaluated on the Middlebury data sets, showing that our approach is among the best (ranked first in the new evaluation system) for all real-time approaches. In addition, since the running time of general BP is linear to the number of iterations, adopting a large number of iterations is not feasible for practical applications. Hence a novel approach is proposed to adaptively update pixel cost. Unlike general BP methods, the running time of our proposed algorithm dramatically converges.

1 Introduction

Stereo vision has traditionally been one of the most extensively investigated topics in computer vision, and is still attracting the attention of many researchers. As a consequence, a variety of approaches have been proposed and an excellent survey of stereo algorithms can be found in [10].

In general, stereo algorithms can be categorized into two major classes: local methods and global methods. Local algorithms, which are based on correlation can have very efficient implementation that are suitable for real-time application [17, 18]. The central problem of local window-based algorithms is how to determine the size and shape of the aggregation window. That is, a window must be large enough to cover sufficient intensity variation while small enough to avoid crossing depth discontinuities for reliable estimation. This inherent ambiguity causes problems such as noisy disparities in textureless region and blurred object boundaries.

Global methods make explicit smoothness assumptions of the disparity map and minimize some cost function. A classic category of global methods is Dynamic Programming

BMVC 2006 doi:10.5244/C.20.101

(DP) based [19, 20]. DP technique can offer optimized solution for independent scanlines in an efficient manner. Due to DP's one dimensional optimization solution and efficient performance, it is the algorithm of choice for many real-time stereo applications [12, 13, 14]. The major problem of DP is that inter-scanline consistency cannot be well enforced, leading to the well-known "streaking" artifacts. Although new algorithms [20, 21] have been proposed to reduce the effect, it can hardly be eliminated. Recently, new global optimization methods such as Belief Propagation (BP) and Graph cut (GC) have attracted much attention. Unlike DP, these methods [8, 11, 6, 4] enforce the optimization in two dimensions, i.e. the entire image. Although some of the most impressive stereo results are obtained, both BP and GC are typically computionally expensive and therefore real-time performance has never been achieved. Recently, Felzenszwalb et al. proposed an efficient BP algorithm [5] uses a hierarchical approach for reducing the complexity. However, it still requires about one second to compute a small image (i.e. 384×288) and cannot achieve real-time performance yet.

In this paper, we propose a real-time belief propagation stereo approach. This algorithm is based on a global energy-minimization framework which contains two terms, the data term and smoothness term. Thus our method can be treated as a two-step algorithm: the construction of the data term and the iterative optimization of the smoothness term. The second step is the essential part of BP, while at the same time, is commonly believed to be the bottleneck of the practical use of the algorithm. Hence, the main contributions of this paper are: first, providing a good accelerator for all the BP based algorithms; second, providing a high quality real-time stereo matching approach.

The rest of this paper is organized as follows. Section 2 gives a description of our stereo matching approach. In Section 3, we propose a fast-converging BP algorithm that greatly reduce the complexity when a large number of iterations are used. Section 4 details how our algorithm is implemented in GPU to gain performance improvement. Our experimental results are presented in Section 5 and in Section 6 we conclude.

2 Approach Description

The algorithm can be partitioned into two blocks: correlation volume computation and hierarchical BP implementation, which correspond to the two terms of the global energy: the data term E_D and the smoothness term E_S respectively.

$$E(d) = E_D(d) + E_S(d); \tag{1}$$

The correlation volume module constructs the data term, and the hierarchical BP module iteratively updates the smoothness term to minimize the global energy.

2.1 Correlation Volume Computation

In the correlation volume computation module, we compute the matching cost in a similar way as the Birchfield and Tomasi's pixel dissimilarity [3], that is for each disparity value, five matching costs are compute. In order to reduce noise, the matching cost is passing through a gaussian filter with σ one pixel. The minimum of the matching costs is selected and compared with a threshold T, multiply the smaller one with a weighting parameter η to get the data term and send it into the hierarchial BP module.

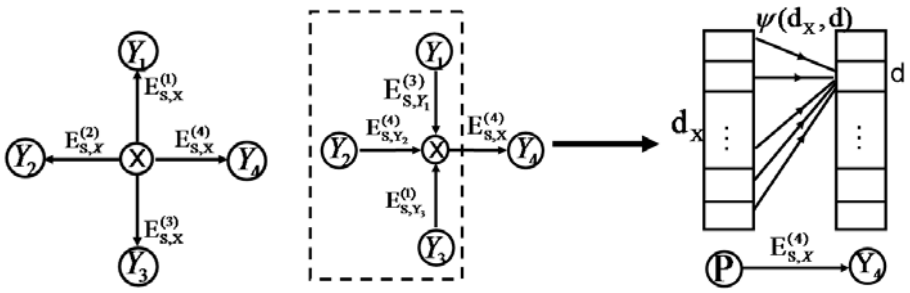

Figure 1: In the left figure, pixel $\mathbf{X}$ has four neighbors $\mathbf{Y}_1$, $\mathbf{Y}_2$, $\mathbf{Y}_3$, $\mathbf{Y}_4$, there's a smoothness term between pixel $\mathbf{X}$ and each of its neighboring pixels. The right figure provides an example showing how to update the smoothness term $E^{(4)}_{S,\mathbf{X}}$ between pixel $\mathbf{X}$ and one of its neighbors $\mathbf{Y}_4$.

2.2 Hierarchical BP

The basic idea of loopy belief propagation algorithm [8] is first gathering information from a pixel's neighbors and incorporate the information to update the smoothness term between the current pixel and its neighboring pixels, and iteratively optimizing the smoothness term to achieve global energy minimization. This is different from the scanline optimization algorithms which only enforce the smoothness along each scanline, because in these algorithms, the smoothness cost information propagates only along the scanline, while in global algorithms like loopy believe propagation and graph cuts algorithms, the smoothness cost information is propagating across the whole image.

Figure 1 provides an example of how to update the smoothness term $E^{(4)}_{S,\mathbf{X}}$ between pixel $\mathbf{X}$ and one of its neighbors $\mathbf{Y}_4$. The first step is using Equation 2 to incorporate the data term of $\mathbf{X}$ ($E_{D,\mathbf{X}}$) with the smoothness term of its other neighboring pixels to generate a new jump cost $M^{(4)}_{S,\mathbf{X}}$. In this paper, we define this cost as **multi-pass jump cost**. The new smoothness term $E^{(4),new}_{S,\mathbf{X}}(d)$ between $\mathbf{X}$ and its neighbor $\mathbf{Y}_4$ is then updated by computing the smallest jump cost using Equation 3.

$$M^{(4)}_{S,\mathbf{X}}(d_\mathbf{x}) = E_{D,\mathbf{X}}(d_\mathbf{x}) + E^{(3),old}_{S,\mathbf{Y}_1}(d_\mathbf{x}) + E^{(4),old}_{S,\mathbf{Y}_2}(d_\mathbf{x}) + E^{(1),old}_{S,\mathbf{Y}_3}(d_\mathbf{x}), \quad (2)$$

$$E^{(4),new}_{S,\mathbf{X}}(d) = arg\min_{d_\mathbf{x}}(M^{(4)}_{S,\mathbf{X}}(d_\mathbf{x}) + \Psi(d_\mathbf{x}, d)), \quad (3)$$

we define $\Psi(d_\mathbf{x}, d)$ as **single-pass jump cost** between two neighboring pixels, it is linear to the absolute difference of the disparities of pixel $\mathbf{X}$ and $\mathbf{Y}_4$. However, in order to increase the robustness to outliers, a threshold λ is added as shown in Equation 4.

$$\Psi(d_\mathbf{x}, d) = \mathbf{min}(\lambda, \rho \mid \mathbf{d_x} - \mathbf{d} \mid), \quad (4)$$

The smoothness term (E_S) is iteratively updated which results in the minimization of the global energy E:

$$E(d) = \sum_{\mathbf{X}} E_\mathbf{X}(d) = \sum_{\mathbf{X}} E_{D,\mathbf{X}}(d) + E^{(3)}_{S,\mathbf{Y}_1}(d) + E^{(4)}_{S,\mathbf{Y}_2}(d) + E^{(1)}_{S,\mathbf{Y}_3}(d) + E^{(2)}_{S,\mathbf{Y}_4}(d), \quad (5)$$

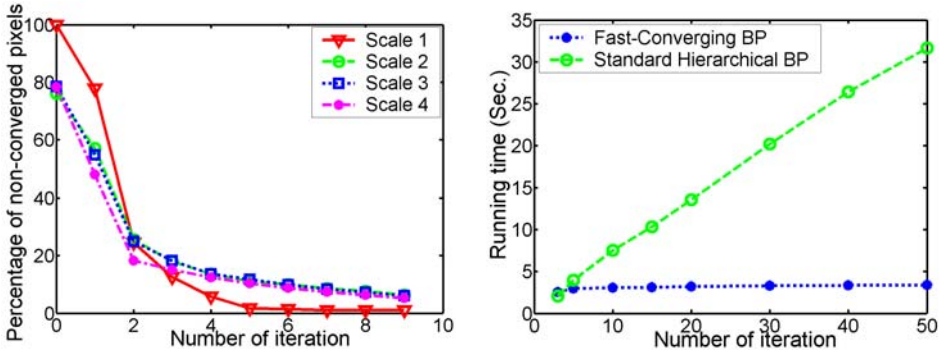

Figure 2: The left figure provides the percentage of the non-converged pixels after every iteration, and the right figure provides the comparison of the running time of fast-converging BP and Standard hierarchical BP algorithms. Both algorithms are run on Tsukuba data set with the same number of iterations on all the four scales.

The global energy converges after a certain number of iterations. The disparity value on each pixel $\mathbf{X}$ is calculated as following:

$$D_{\mathbf{X}} = arg\min_{d}(E_{\mathbf{X}}(d)), \tag{6}$$

The general loopy belief propagation algorithm is too slow to be practically used while achieving very good result, not only because the algorithm itself is complicated, but also because a certain number of iterations are required before the algorithm converges. Felzenszwalb [5] provides a hierarchical algorithm which runs much faster than the previous algorithms while yielding comparable accuracy. The main difference between the hierarchical BP and general BP is that hierarchical BP works in a coarse-to-fine manner, first performing BP at the coarsest scale, then using the output from the coarser scale to initialize the input for the next scale.

3 Fast-Converging BP

For standard BP algorithms, in order to achieve the best stereo results, a large number of iterations are required to guarantee the convergence. However, since the running time is linear to the number of iterations, large number of iterations will greatly hurt the practical application of BP algorithms.

Actually, there are lots of redundant computations involved in standard BP. In essence, by only updating pixels that have not yet converged, fast-converging BP removes those redundant computations while achieving the same accurancy as standard BP.

In detail, the new smoothness term of one of the pixels in the graph is updated according to its own data term ($E_{D,\mathbf{X}}$), the previous smoothness term of its four neighboring pixels ($E_{S,\mathbf{Y}_i}^{(i),old}$), and the **single-pass jump cost** function Ψ. Since the data term and the **single-pass jump cost** stay unchanged, they can be treated as fixed parameters, and the updated smoothness term ($E_{S,\mathbf{X}}^{(i),new}$) of a pixel $\mathbf{X}$ in the graph thus becomes a function of

variables containing only the previous smoothness term of its four neightboring pixels, for instance:

$$E_{S,\mathbf{X}}^{(4),new} = f(E_{S,\mathbf{Y}_1}^{(3),old}(d_{\mathbf{x}}), E_{S,\mathbf{Y}_2}^{(4),old}(d_{\mathbf{x}}), E_{S,\mathbf{Y}_3}^{(1),old}(d_{\mathbf{x}})), \tag{7}$$

As a result, before updating the smoothness term of a pixel $\mathbf{X}$ at iteration i, check whether the smoothness term of its four neighboring pixels at iteration $i-1$ and at iteration $i-2$ are equivalent or not. If the smoothness terms are the same, it is not necessary to update the smoothness term of pixel $\mathbf{X}$.

Figure 2 shows that after several numbers of iterations, most of the pixels on the graph converge. The fast-converging BP algorithm thus ignores these pixels, the updating scheme is only applied to the non-converged pixels, which greatly decrease the running time of BP approaches with large number of iterations. Figure 2 also shows that the running time of the standard BP is linear to the number of iterations while the running time of the fast-converging BP dramatically converges.

We have successfully implemented this fast-converging BP approach on CPU, and are looking forward to implementing it on GPU. The experiment results provided in Figure 2 are based on CPU implementation.

4 GPU Implementation

We have implemented both the first (correlation volume computation) and second step (hierarchical BP) on graphics hardware to facilitate real-time computation. The GPU implementation of the first step is very simple, so we only focus on the second step.

Algorithm 1 Updating the Smoothness Term on GPU

Require: $E_{S,old}^{(i),coarse}$, $i = 1,2,3,4$.

1: Initialize $E_{S,old}^{(i),fine}$: $E_{S,old}^{(i),fine}(X) = E_{S,old}^{(i),coarse}(X/2)$.
2: Initialize N with the number of iterations of the current scale.
3: **repeat**
4: -Compute $M_S^{(i)}$ according to Equation 2;
5: -Compute the minimun of $M_S^{(i)}$ for each pixels, plus it with the threshold λ which is provided in Equation 4, save it as MIN_S;
6: -Update $E_{S,new}^{(i)}$:
for d **from** 1 **to** $NR_{disp}-1$:
$M_S^{(i)}(d)$=**min**$(M_S^{(i)}(d), M_S^{(i)}(d-1)+\rho)$;
for d **from** $NR_{disp}-2$ **to** 0:
$M_S^{(i)}(d)$=**min**$(M_S^{(i)}(d), M_S^{(i)}(d+1)+\rho, MIN_S)$;
$E_{S,new}^{(i)} = M_S^{(i)}$;
7: -$E_{S,old}^{(i)} = E_{S,new}^{(i)}$;
8: -Normalize $E_{S,new}^{(i)}$, such that $\sum_{d=0}^{N-1} E_{S,new}^{(i)}(d) = 0$
9: -Decrease N by 1.
10: **until** $N \leq 0$.

In our implementation, there are four scales in the hierarchical BP, and the main process for each scale are the same. The updating scheme for each scale are summarized in Algorithm 1. For each scale, eight textures are used to store the smoothness term. The old smoothness term generated in the previous iteration is stored in four of the textures ($E^{(i)}_{S,old}$, $i = 1,2,3,4$), the other four textures are used to store the updated smoothness term ($E^{(i)}_{S,new}$). For the coarsest scale, before the iteration begins, initialize $E^{(i)}_{S,old}$ with all zeros, as to the other scales, $E^{(i)}_{S,old}$ is initialized as Algorithm 1 describes. At the beginning of each iteration, compute **multi-pass jump cost** $M^{(i)}_S$ from the data term and the previous smoothness term as described in the Approach Section. The next step is to update the smoothness term using Equation 3. The complexity of this problem is $O(NR^2_{disp})$ (NR_{disp} is the number of disparity levels), but the updating scheme provided in Algorithm 1 reduces the complexity to $O(N)$. Finally, normalize $E^{(i)}_{S,new}$, such that $\sum_{d=0}^{N-1} E^{(i)}_{S,new}(d) = 0$.

When the iteration is completed in the fine scale, uses Equation 5 and 6 to create the disparity map.

5 Experimental Results

We tested our **real-time BP** algorithm on a 3 GHz PC running Direct3D 9.0. The GPU is a Geforce 7900 GTX graphics card with 512M video memory from NVIDIA. All shaders are implemented using HLSL and complied using pixel shader 3.0 profile. The following experiments are conducted to evaluate both the quality and efficiency performance of our algorithm. The same parameter settings were used throughout the experiments. Two parameters $T = 30$ and $\eta = 0.15$ are used in the correlation volume computation module, another two parameters are involved in the calculation of the **single-pass jump cost**: ρ and λ. ρ is set to 1.0, and λ is determined by the the number of disparity levels (NR_{disp}) of the input data set: $\lambda = (2.0 \times NR_{disp})/16$. 16 is the number of disparity levels of the Tsukuba data set. In this paper, we implement hierarchical BP in GPU with four scales, and the typical iterations for each scale are $(5,5,10,4)$, from coarse-to-fine scale.

5.1 Quality Evaluation with Ground Truth

We first evaluate the reconstruction quality of our approach using the benchmark Middlebury stereo data set based on known ground truth. The new evaluation test data consists of four stereo pairs within which "Tsukuba" and "Venus" are standard stereo data with slanted surfaces and up to 20 disparity levels, "Teddy" and "Cones" are both new adopted image pairs with more complicated scene structure and much larger disparity ranges. We evaluate the numerical accuracy of the dense disparity maps generated by our algorithm using the online system at [7]. The results from all test images are shown in figure 3.

This quantitative evaluation confirms that, as demonstrated in Table 1, our **real-time BP** performs as well as other global optimization approaches. Generally speaking, the overall performance is ranked between the best belief propagation based algorithms which are the current state-of-the-art stereo algorithms and the Graph Cuts based algorithms. One thing worth noticing is that most of these methods, such as [1, 11, 6], integrate multiple low-level visual cues (e.g., segmentation, edges, visibility testing) as either soft or hard constraints to improve stereo matching while our approach works under a basic and

Algorithm	Avg. Rank	Tsukuba nonocc	Tsukuba all	Venus nonocc	Venus all	Teddy nonocc	Teddy all	Cones nonocc	Cones all
DoubleBP [1]	2.3	0.88[1]	1.29[1]	0.14[2]	0.60[5]	3.55[1]	8.71[3]	2.90[3]	9.24[4]
SymBP+occ [11]	5.1	0.97[2]	1.75[5]	0.16[3]	0.33[2]	6.47[6]	10.7[4]	4.79[11]	10.7[8]
Our Algorithm	**10.4**	**1.49**[10]	**3.40**[13]	**0.77**[7]	**1.90**[11]	**8.72**[13]	**13.2**[8]	**4.61**[9]	**11.6**[10]
GC+occ [6]	11.5	1.19[4]	2.01[9]	1.64[14]	2.19[13]	11.2[16]	17.4[16]	5.36[13]	12.4[13]
MultiCamGC [9]	12.0	1.27[5]	1.99[8]	2.79[19]	3.13[17]	12.0[17]	17.6[17]	4.89[12]	11.8[11]
GC [10]	16.6	1.94[12]	4.12[15]	1.79[16]	3.44[18]	16.5[21]	25.0[22]	7.70[17]	18.2[18]
Belief prop. [8]	NA	1.15	NA	0.98	NA	NA	NA	NA	NA
HierarchicalBP [5]	NA	1.86	NA	0.96	NA	NA	NA	NA	NA

Table 1: Performance comparison of the proposed method with other high-quality global optimization approaches. This measure is computed for three subsets of the images,they are ”nonocc”: the subset of non-occluded pixels, ”all”: pixels that are either non-occluded or half-occluded. The subscript is the relevant rank of each item on the table. Note that since the old Middlebury table which contains several bp based stereo methods is no longer functional, we have collected the non-occluded (overall) error rate of the shared test data ’Tsukuba’ and ’Venus’. Those numbers that are not available due to this reason are labeled ”NA”.

clean probabilistic framework without any additional information incorporated. Moreover, the iteration numbers used across all experiments is only 4. Although increasing the number of iterations can produce stronger results simultaneously, we balance the quality and efficiency by not using too may iterations and our later experiments will show this compensation does not prevent us from achieving satisfying results.

In addition, in terms of accuracy, Table 2 shows that our **real-time BP** outperforms all the other methods that can achieve real-time or near real-time performance listed on the new Middlebury evaluation table. Since the old evaluation table at Middlebury, which contains some algorithms that aim real-time, is no longer functional, we collecte the non-occluded error percentage of the shared test data ’Tsukuba’ and ’Venus’ and provide results in Table 2 for reference.

Algorithm	Avg. Rank	Tsukuba nonocc	Venus nonocc	Teddy nonocc	Cones nonocc
Our Algorithm	**10.4**	**1.49** [10]	**0.77** [7]	**8.72** [13]	**4.61** [9]
RealTimeGPU [12]	14.3	2.05 [14]	1.92 [17]	7.23 [8]	6.41 [15]
ReliabilityDP [13]	15.6	1.36 [7]	2.35 [18]	9.82 [15]	12.9 [22]
Realtime [16]	NA	4.25 [NA]	1.32 [NA]	NA	NA
Realtime DP [14]	NA	2.85 [NA]	6.25 [NA]	NA	NA
Max. surf. [15]	NA	11.10[NA]	5.51 [NA]	NA	NA

Table 2: Performance comparison of the proposed method with other real-time approaches listed on the Middlebury evaluation tables.

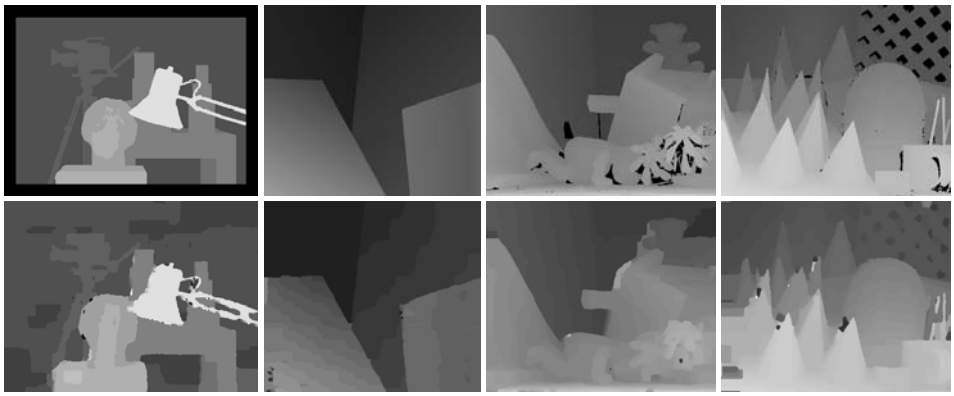

Figure 3: Resulting disparity maps from the Middlebury stereo data set. (top row) Ground truth; (bottom row) Disparity maps generated from our method.

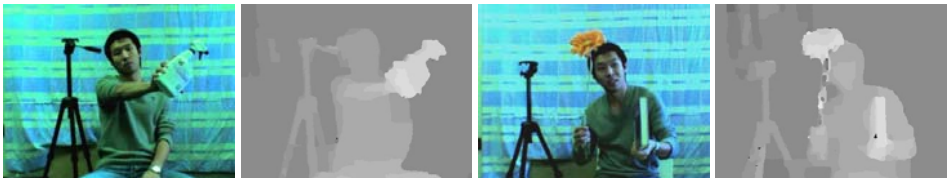

Figure 4: Two sample images and their corresponding disparity maps from our live system on a 3 GHz PC with a Geforce 7900 GTX graphics card from NVIDIA. Our system can reach 16 fps given 320×240 input images and 16 disparity levels.

5.2 Live System and Efficiency Performance

We integrated our algorithm into a real-time stereo system with live video input. The stereo pairs are rectified and with lens distortion removed. This pre-processing is implemented in the GPU using texture-mapping functions. Figure 4 shows the results of applying our **real-time BP** algorithm to some live images captured from our system. These real scene images are with resolution 320×240 and 16 disparity levels. Note that the scene structures and object borders have been well detected. The speed is about 16 fps for our live system.

To further evaluate the efficiency performance of our algorithm, we test our system against the four Middlebury test data under different configurations and summarize the results in Table 3. Two characteristics of our **real-time BP** algorithm can be observed from the measurements. First, by utilizing graphics hardware acceleration, we can achieve a speedup factor up to 45 compared to its CPU counterpart. Second, the error percentage changed slightly with the increasing of iterations. Using a few iterations are able to produce strong results. These two characteristics cooperatively explain why our algorithm is very suitable for real-time application.

Iteration (N)	MDE/Second		Error(%)				Avg. Rank
	CPU	GPU	Tsukuba	Venus	Teddy	Cones	
2	0.49	22.2	1.59	0.90	8.89	4.73	11.0
4	0.39	17.0	1.49	0.77	8.72	4.61	10.4
6	0.31	13.7	1.47	0.67	8.68	4.57	9.8
10	0.23	10.1	1.47	0.60	9.09	4.54	9.7

Table 3: Running time evaluation on the four new Middlebury stereo data. The speed and overall error rate corresponding to different number of iterations are presented in the table. Speed performance is measured by million disparity estimations per second (MDE/s). Here both the CPU and GPU's MDE/s values are calculated based on Tsukuba data set. Clearly GPU acceleration can achieve a high speedup factor compared to the CPU implementation. In addition, iterations used for each scale are $(5, 5, 10, N)$, from coarse-to-fine scale. N is the variable provided in the table.

6 Conclusions and Future Work

In this paper, a real-time stereo model based on hierarchical belief propagation was proposed, which demonstrates that global optimization based stereo matching is possible for real-time applications. The whole algorithm design in this paper is very clean and results in very high quality stereo matching. We qualified the accuracy of the stereo results using the Middlebury benchmark, which shows that our algorithm outperforms all the other real-time stereo algorithms.

Looking into the future, both the quality and the speed of the proposed real-time BP approach can be improved. For the quality, more constraints and priors (e.g. edges, corners, junctions, segmentation, visibility) can be incorporated; for the speed, in Section 3, we have proposed an approach which allows large number of iterations to guarantee the convergence, for instance, the running time of fast-converging BP with 100 iterations is even less than the running time for standard BP with 5 iterations. We're planning to transfer it to GPU in the near future. In addition, [5] presented an approach which can decrease both the storage requirements and the running time by a factor of two. Because for a bipartite graph, all the nodes can be separated into two clusters, for each iteration, only one cluster's smoothness term needs to be updated. We have implemented this approach on CPU, and we would also like to implement this approach on GPU to improve the speed upto a factor of two in the near future.

References

[1] Q. Yáng, L. Wang, R. Yang, H. Stewénius. and D. Nistér, Stereo matching with color-weighted correlation, hierarchical belief propagation and occlusion handling, *CVPR*, June 2006.

[2] M. Bleyer and M. Gelautz, A Layered Stereo Algorithm Using Image Segmentation and Global Visibility Constraints, *ICIP*, pp. 2997-3000, 2004.

[3] S. Birchfield and C. Tomasi, A Pixel Dissimilarity Measure That Is Insensitive to Image Sampling, *PAMI*, Vol. 20, No. 4, April 1998.

[4] Y. Boykov, O. Veksler and R. Zabih, Fast Approximate Energy Minimization via Graph Cuts, *PAMI*, Vol. 23, No. 11, 2001.

[5] P. F. Felzenszwalb and D. P. Huttenlocher, Efficient Belief Propagation for Early Vision, *CVPR*, Vol. I:261-268, 2004.

[6] V. Kolmogorov and R. Zabih, Computing Visual Correspondence with Occlusions using Graph Cuts, *ICCV*, Vol. I:508-515 2001.

[7] D. Scharstein and R. Szeliski, Middlebury Stereo Vision Research Page, *http://bj.middlebury.edu/~schar/stereo/newEval/php/results.php*

[8] J. Sun, N.-N. Zheng and H.-Y. Shum, Stereo Matching Using Belief Propagation, *PAMI*, Vol. 25, No. 7, July 2003.

[9] V. Kolmogorov and R. Zabih, Multi-camera scene reconstruction via graph cuts, *ECCV*, Vol. III:82-96, 2002.

[10] D. Scharstein and R. Szeliski, A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms, *IJCV*,47(1):7-42, May 2002.

[11] J. Sun, Y. Li, S. B. Kang and H.-Y. Shum, Symmetric Stereo Matching for Occlusion Handling, *CVPR*, Vol. II:399-406, 2005.

[12] L. Wang, M. Liao, M. Gong, R. Yang, and D. Nistér, High-quality real-time stereo using adaptive cost aggregation and dynamic programming, 3DPVT 2006.

[13] M. Gong and Y. Yang, Near real-time reliable stereo matching using programmable graphics hardware, *CVPR*, Vol I:924-931, 2005.

[14] S. Forstmann, J. Ohya, Y. Kanou, A. Schmitt, and S. Thuering, Real-time stereo by using dynamic programming, *CVPR*, Workshop on real-time 3D sensors and their use, 2004.

[15] C. Sun, Fast stereo matching using rectangular subregioning and 3D maximum-surface techniques, *CVPR*, Stereo workshop, 2001.

[16] H. Hirschmüller, Improvements in real-time correlation-based stereo vision, *CVPR*, Stereo workshop, 2001.

[17] H. Hirschmüller, P. R. Innocent, and J. M. Garibaldi. Real-time correlation-based stereo vision with reduced border errors, *IJCV*, 47(1/2/3):229-246, April-June 2002.

[18] O. Veksler. Fast variable window for stereo correspondence using integral images, *CVPR*, Vol I: 556-561, 2003.

[19] Yuichi Ohta, Takeo Kanade, Stereo by intea nad inter-scanline search using dynamic programming, *PAMI*, 1985.

[20] J. Kim, K.M. Lee, B.T. Choi, and S.U. Lee. A dense stereo matching using two-pass dynamic programming with generalized ground control points, *IEEE CVPR*, Vol II: 1075-1082, 2005.

[21] O. Veksler. Stereo correspondence by dynamic programming on a tree, *IEEE CVPR*, Vol II: 384-390, 2005.