Approximate Belief Propagation by Hierarchical Averaging of Outgoing Messages

Koichi Ogawara
Institute of Advanced Study, Kyushu University, Fukuoka, JAPAN
Email: ogawara@ait.kyushu-u.ac.jp

***Abstract*—This paper presents an approximate belief propagation algorithm that replaces outgoing messages from a node with the averaged outgoing message and propagates messages from a low resolution graph to the original graph hierarchically. The proposed method reduces the computational time by half or two-thirds and reduces the required amount of memory by 60% compared with the standard belief propagation algorithm when applied to an image.**

The proposed method was implemented on CPU and GPU, and was evaluated against Middlebury stereo benchmark dataset in comparison with the standard belief propagation algorithm. It is shown that the proposed method outperforms the other in terms of both the computational time and the required amount of memory with minor loss of accuracy.

***Keywords*-belief propagation; stereo; GPU;**

I. Introduction

Many low-level vision problems such as disparity estimation in stereo [1] and image segmentation [2] can be formulated by markov random field (MRF). They are known as NP-hard problems [3] and a suboptimal solution, or the optimal solution in some cases, can be obtained via a global optimization framework such as belief propagation (BP) [4] or graph-cut [5]. Except for some special cases [6], [7], it is equivalent to solving a multi-label assignment problem on a loopy graph that requires iterative computations, thus they have been considered not to be suitable for real-time applications.

The computational complexity of a standard max-product, or min-sum, belief propagation algorithm is $O(NL^2T)$ where N is the number of nodes, or pixels, L is the number of labels and T is the number of iterations [8]. To reduce the computational time, several methods have been proposed such as linear time computation of messages [4], hierarchical belief propagation [4], the use of parallel processors on Graphics Processing Unit (GPU) [9], etc.

The challenge in implementing the algorithm on GPU, or embedded systems, is to reduce the required amount of memory because of the bottleneck in memory bandwidth or the limitation of available memory. For this, several methods have been proposed such as predictive coding of messages [10], tile-based belief propagation [11]! $constant space belief propagation [12], etc.

However, all the above mentioned methods use the standard algorithm to compute messages without modification. In contrast, the proposed method simplifies that algorithm to reduce the computational time by half or two-thirds and to reduce the required amount of memory by 60% compared to the standard method when applied to an image with minor loss of accuracy. The major contribution of the proposed method is two-fold: (1) outgoing messages from a node are replaced with the averaged outgoing message to reduce both the computational time and the required amount of memory drastically, (2) averaged outgoing messages are propagated from a low resolution graph to the original graph hierarchically to avoid loss of accuracy caused by the simplification.

The proposed method was evaluated against Middlebury stereo benchmark dataset [13] in comparison with the standard belief propagation algorithm. The proposed method can be easily parallelized, thus it was also implemented and evaluated on GPU.

II. Belief Propagation

Problems such as disparity estimation in stereo or image segmentation can be formulated as to assign a label $\{1, \ldots, L\}$ to each pixel in an image given an observation $\boldsymbol{Y} = (\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_N)$. The optimal label assignment $\boldsymbol{X} = (x_1, x_2, \ldots, x_N)$ can be obtained by minimizing

$$E(\boldsymbol{X}) = \sum_{i \leq N} D_i(x_i) + \sum_{(i,j)} V(x_i, x_j) \tag{1}$$

where (i, j) is a set of neighboring nodes in a 4-connected graph where a node represents a pixel in the image. $D_i(x_i)$ is a cost function that evaluates x_i assigned to node i where $\boldsymbol{y}_i$ is observed. $V(x_i, x_j)$ is a penalty function that evaluates x_i, x_j assigned to neighboring nodes i, j respectively. A penalty is usually imposed when different labels are assigned to neighboring nodes, thus $V(x_i, x_j)$ is also called a smoothness term.

In loopy belief propagation, eq. (1) is minimized by iteratively propagating messages between neighboring nodes. A message $\boldsymbol{m}^t_{i \to j}$ from node i to j at time t is represented by a L-elements array and x_j-th element is computed as

$$\begin{aligned} &\boldsymbol{m}^t_{i \to j}[x_j] \\ &= \min_{x_i} \left(V(x_i, x_j) + D_i(x_i) + \sum_{s \in N_e(i) \setminus j} \boldsymbol{m}^{t-1}_{s \to i}[x_i] \right) \end{aligned} \tag{2}$$

where $N_e(i)$ is a set of nodes adjacent to node i. At $t = 0$, all the elements are initialized to 0.

1051-4651/10 $26.00 © 2010 IEEE
DOI 10.1109/ICPR.2010.338

After all incoming messages to node i converge, the label x_i^* at node i is estimated as

$$x_i^* = \underset{x_i}{\operatorname{argmin}} \left(D_i(x_i) + \sum_{j \in N_e(i)} \boldsymbol{m}_{j \to i}^T[x_i] \right). \quad (3)$$

III. Proposed Method

A. Averaging of Outgoing Messages

As shown in the third term of the right-hand side of eq. (2), an outgoing message from a node is computed based on the sum of incoming messages from the adjacent nodes except the one in the outgoing direction.

In the proposed method, this third term is replaced with the sum of all incoming messages multiplied by $\frac{\#N_e(i)-1}{\#N_e(i)}$ as

$$\boldsymbol{m}_{i \to j}^t[x_j] = \min_{x_i} \left(V(x_i, x_j) + D_i(x_i) + \frac{\#N_e(i)-1}{\#N_e(i)} \sum_{s \in N_e(i)} \boldsymbol{m}_{s \to i}^{t-1}[x_i] \right) \quad (4)$$

where $\#N_e(i)$ is the total number of nodes adjacent to node i. In case of a 4-connected graph, $\#N_e(i)$ is 4.

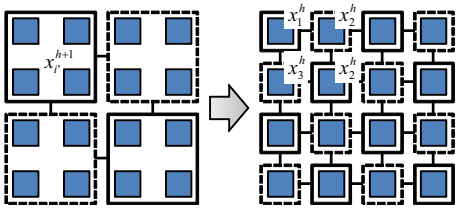
With this new algorithm, all the outgoing messages from a node become identical to each other, thus the number of message computation per node is reduced to 1 while it takes $\#N_e(i)$ times per node with the conventional algorithm. This reduction leads to the reduction of the entire computational time by half or two-thirds in case of a 4-connected graph as shown later. Furthermore, the new algorithm only requires memory for a single outgoing message per node, while the conventional algorithm requires memory for $\#N_e(i)$ outgoing messages. Thus, the total required amount of memory becomes $\frac{1+1}{1+\#N_e(i)}$, i.e. 40% in case of a 4-connected graph, if the amount of memory required for the data cost $D_i(x_i)$ in eq. (4) is the same as that for a single outgoing message.

The label x_i^* at node i is estimated from eq. (3) in the same way.

B. Hierarchical Propagation of Averaged Outgoing Messages

When messages are initialized to 0 and the method proposed in the previous section is applied, messages in many nodes converge to local solutions as shown later.

To solve this problem, hierarchical belief propagation [4] is employed. This method is primarily used to propagate messages over long distances in a small number of iterations. Here, it is used to prevent averaged outgoing messages from falling into local solutions. In the proposed method, the approximate belief propagation algorithm explained in the previous section is iterated in a lower resolution graph as shown in Figure 1 (a) until the messages converge well. Then, the convergent messages are used as the initial messages in a higher resolution graph as shown in Figure 1 (b) and the approximate belief propagation algorithm is iterated again until the messages converge well in the same way. As a consequence, initial messages in a higher resolution graph are getting closer to the optimal messages and the chance of being trapped into local solutions is reduced.



(a) Low resolution graph (b) High resolution graph

Figure 1. Hierarchical propagation of averaged outgoing messages.

Hierarchical propagation of averaged outgoing messages proceeds as follows. Firstly, the data costs in eq. (4) in lower resolution graphs are calculated as

$$D_{i'}^{h+1}(x_{i'}^{h+1}) = \sum_{i \in [1,4]} D_i^h(x_i^h)$$

where $h \in [1, H]$ represents the level of hierarchy. $h = 1$ means the original highest resolution graph, and the resolution of a graph is reduced by half as the level is increased by 1.

Then, the approximate belief propagation algorithm is iterated in each graph from the lowest resolution to the original resolution, and the convergent messages in each graph are propagated to higher graphs as

$$\boldsymbol{m}_{i \to j}^{t,h}[x_i^h] = \boldsymbol{m}_{i' \to j'}^{t,h+1}[x_{i'}^{h+1}] \quad i \in [1,4], j' \in N_e(i')$$

where node j at level h is the adjacent node in the same direction from node i' to node j' at level $h+1$.

Note that messages of the nodes surrounded by a solid line and those surrounded by a dashed line are alternatively calculated and propagated so as to guarantee the consistent access to the variables when computed in parallel on GPU.

IV. Experiment

In this section, the proposed method was evaluated against Middlebury stereo benchmark dataset [13] in comparison with the standard belief propagation algorithm in terms of the accuracy of disparity estimation.

Then, we discuss the reason why the proposed method works well based on the analysis of the experimental results.

Finally, the proposed method was implemented on CPU and GPU, and the rate of reduction of the computational time and the required amount of memory was evaluated.

A. Experimental Conditions

In this experiment, the data function in eq. (1) is defined as

$$D_i(x_i) = \min(|I_L(i) - I_R(i - x_i)|, \tau) \tag{5}$$

where $I_L(i)$ means the i-th pixel value in the left image and $I_R(i - x_i)$ means the $(i - x_i)$-th pixel value in the right image. τ is a constant to suppress the effect of outliers and we use $\tau = 30$ in this experiment.

The smoothness term in eq. (1) is defined as a linear function of the difference between labels as

$$V(x_i, x_j) = \min(c|x_i - x_j|, d) \tag{6}$$

where d is a constant to suppress penalty at the occluding boundary of regions where label changes largely and we use $c = 14.0\,!\,\$d = 33.6$ in this experiment.

By defining the smoothness term as eq. (6), the computational complexity of message computation becomes linear in the number of labels without affecting the result as explained below. With this method, the total computational complexity is reduced to $O(NLT)$, however the proposed method reduces the number of message computations, thus the rate of reduction of the computational time becomes small if the proposed method is applied based on this method. However, this method is widely used as an efficient technique, so we use this technique for fair comparison.

1) Linear Time Computation of Messages [4]: In eq. (2), for each element of a message $\boldsymbol{m}^t_{i\to j}$ from node i to j, x_i that minimizes the element is chosen from L labels, thus the total computational complexity is $O(L^2)$.

Eq. (2) can be re-written as

$$\boldsymbol{m}^t_{i\to j}[x_j] = \min_{x_i}(V(x_i, x_j) + \boldsymbol{h}^t_{i\to j}[x_i]), \tag{7}$$

$$\boldsymbol{h}^t_{i\to j}[x_i] = D_i(x_i) + \sum_{s\in N_e(i)\setminus j} \boldsymbol{m}^{t-1}_{s\to i}[x_i]. \tag{8}$$

If $V(x_i, x_j)$ is defined as eq. (6), messages can be computed in linear time as follows. Hereafter, we use $\boldsymbol{m}[x_j]$, $\boldsymbol{h}[x_i]$ instead of $\boldsymbol{m}^t_{i\to j}[x_j]$, $\boldsymbol{h}^t_{i\to j}[x_i]$ if not ambiguous.

1) Initialization
   for x_j **from** 1 **to** L :
   $\boldsymbol{m}[x_j] \leftarrow \boldsymbol{h}[x_j]$
2) Forward computation
   for x_j **from** 2 **to** L :
   $\boldsymbol{m}[x_j] \leftarrow \min(\boldsymbol{m}[x_j], \boldsymbol{m}[x_j - 1] + c)$
3) Backward computation
   for x_j **from** $L - 1$ **to** 1 :
   $\boldsymbol{m}[x_j] \leftarrow \min(\boldsymbol{m}[x_j], \boldsymbol{m}[x_j + 1] + c)$
4) Saturation
   for x_j **from** 1 **to** L :
   $\boldsymbol{m}[x_j] \leftarrow \min(\boldsymbol{m}[x_j], \min_{x_i} \boldsymbol{h}[x_i] + d)$

Table I
ERROR IN DISPARITY ESTIMATION: H MEANS HIERARCHICAL METHOD, SBP MEANS THE STANDARD BELIEF PROPAGATION ALGORITHM AND AOM MEANS THE APPROXIMATED BELIEF PROPAGATION ALGORITHM BY AVERAGING OF OUTGOING MESSAGES.

Algorithm	# iter.	Venus		Cones	
		nonocc [%]	all [%]	nonocc [%]	all [%]
H4+SBP	80	0.95	2.06	5.36	13.56
	320	0.99	2.10	5.28	13.47
H4+AOM	80	1.62	2.79	5.73	14.21
	320	1.62	2.81	5.68	14.13
SBP	80	1.37	2.50	5.06	13.33
	320	1.35	2.46	5.09	13.37
AOM	80	8.58	9.92	10.13	19.26
	320	7.69	9.03	9.76	18.94
Without BP	0	66.32	66.87	69.62	72.95

To apply this method to eq. (4), we only need to replace eq. (8) with

$$\boldsymbol{h}^t_{i\to j}[x_i] = D_i(x_i) + \frac{\#N_e(i) - 1}{\#N_e(i)} \sum_{s\in N_e(i)} \boldsymbol{m}^{t-1}_{s\to i}[x_i]. \tag{9}$$

B. Error in Disparity Estimation

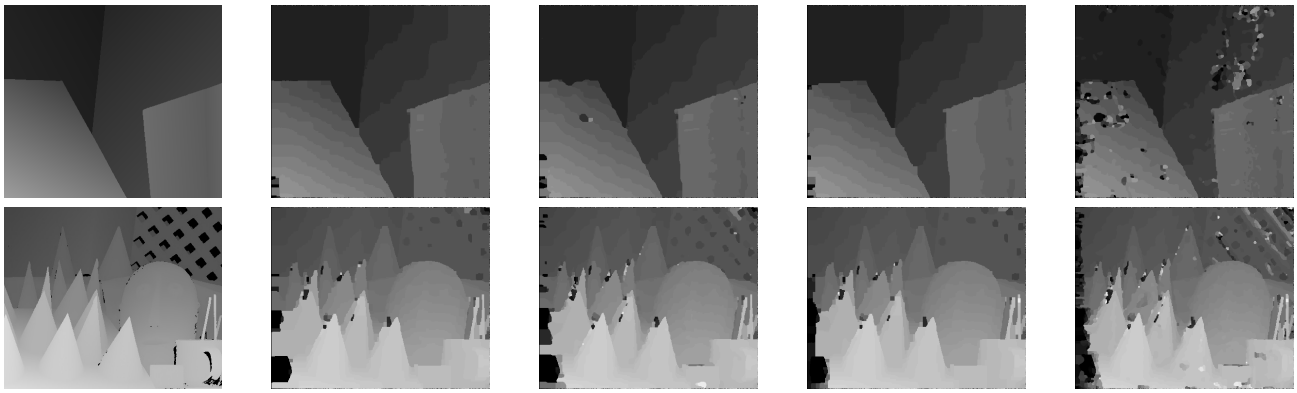
In this section, the proposed method was evaluated against Venus and Cones datasets in Middlebury dataset [13] in comparison with the standard belief propagation algorithm in terms of the accuracy of disparity estimation. The 2 methods were evaluated with and without hierarchical method, and 4 levels were used when hierarchical method was applied. Hereafter, H# means Hierarchical method and the number of levels, SBP means the Standard Belief Propagation algorithm and AOM means the approximated belief propagation algorithm by Averaging of Outgoing Messages. In addition, the result without belief propagation is shown where only the data cost in eq. (1) was evaluated locally.

Table I shows the error in disparity estimation. Following [1], if the difference between the estimated disparity and the ground truth is more than 1, it is regarded as an error. "All" in the Table means the ratio [%] of the number of errornous pixels over the entire pixels. "Nonocc" means the same ratio calculated for non-occluded pixels. The accuracy of H4+SBP and SBP are almost the same and that of H4+AOM is slightly worse, while that of AOM is considerably bad. Figure 2 shows the estimated disparity map. We can see that the error of AOM is clearly large.

C. Discussion

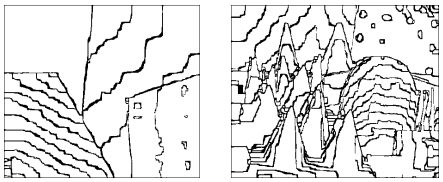
In this section, we discuss why the result of H4+AOM is good, while the result of AOM is not good.

Figure 3 shows the variance of the final outgoing messages of H4+SBP (the total number of iterations is 80) where the black pixels mean the variance is more than 0.1 and the white pixels mean otherwise. This reveals that the final outgoing messages from a node are identical to each other for $85 \sim 90\%$ of the entire nodes when the standard belief



(a) Ground truth (b) H4+SBP (c) **H4+AOM** (d) SBP (e) AOM

Figure 2. Estimated disparity map against Middlebury dataset: top) Venus and bottom) Cones.



(a) Venus (b) Cones

Figure 3. Variance of the final outgoing messages: a black pixel means variance > 0.1 and a white pixel means otherwise.

propagation algorithm is applied. From Figure 2 (a), we can also see that the nodes of large variance correspond to the boundary of regions where neighboring nodes have different labels.

In general, the total number of nodes at the boundary of regions is smaller than the total number of nodes inside regions where neighboring nodes have the same label. So, if (1) outgoing messages of the majority of nodes inside regions converge to the optimal solutions and (2) the labels of the nodes at the boundary of regions do not change largely, the outgoing messages of the nodes at the boundary of regions are smoothed by the propagated correct messages from both side of the boundary, thus they are expected to converge to the near optimal solutions.

Since the latter condition depends on the property of an observation, we focus on the former condition. In conventional belief propagation, messages are initialized to 0 and are updated based on eq. (2) until they converge. In case of a loopy graph, eq. (2) is still correct locally and messages converge to near optimal solutions in most cases. However, eq. (4) is not correct anymore even locally and messages in many nodes converge to local solutions as shown in Figure 4 (e).

Figure 4 (a) shows the error in disparity estimation of H4+SBP and Figure 4 (b)-(e) show the error in disparity estimation of H#+AOM. Black nodes represent wrong estimation, while white pixels represent correct estimation. "Variance" in the Figure means the mean squared difference between the initial message at the final hierarchy and the optimal message calculated over the nodes corresponding to the white pixels in Figure 3 (a). In other words, it shows how much close the initial message and the optimal message are. "Error ratio" in the Figure means the ratio of the wrongly estimated nodes at the end of iterations.

From Figure 4, the larger the number of hierarchies is, the smaller "variance" becomes, that means the closer the difference between the initial message and the optimal message becomes. At the same time, "error ratio" becomes small, that means the number of the wrongly estimated nodes becomes small. In other words, hierarchical method propagates averaged messages to higher hierarchies while managing both avoidance from local solutions and convergence to the optimal solutions. We can see that this behavior contributes to minor loss of accuracy of the proposed method. In all cases, messages in the lowest hierarchy are initialized to 0 and the number of iterations in each hierarchy is 20.

D. Rate of Reduction of the Computational Time and the Required Amount of Memory

In this section, the proposed method was implemented on GPU as well as on CPU, and the rate of reduction of the computational time and the required amount of memory was evaluated. The algorithms were implemented on CPU (Xeon 3.0Ghz, Memory 2G) using Visual Studio 2008 (C++), while they were also implemented on GPU (GeForce GTX285) using CUDA [14].

Table II shows the computational time ([ms]) of 4 algorithms against 2 datasets where the total number of iterations is 80 for all the trials. "Mem" in the table represents the ratio of the required amount of memory based on that of H4+SBP.

In both datasets, algorithms on GPU run $40 \sim 50$ times faster than those on CPU. The fastest algorithm is H4+AOM (GPU). As for the required amount of memory, H4+AOM is the second smallest after AOM. However the accuracy of

variance=1.49 error ratio=0.28 (a) H4+SBP | variance=2.81 error ratio=0.64 (b) H4+AOM | variance=4.04 error ratio=0.72 (c) H3+AOM | variance=10.74 error ratio=3.70 (d) H2+AOM | variance=80.27 error ratio=10.12 (e) H1+AOM

Figure 4. Error in disparity estimation in Venus dataset: a black pixel means a wrongly estimated node and a white pixel means a correctly estimated node. Variance means the mean squared difference between the initial message at the final hierarchy and the optimal message. Error ratio means the ratio of the wrongly estimated nodes.

Table II
THE REQUIRED AMOUNT OF MEMORY AND COMPUTATIONAL TIME.

Algorithm	Mem	Venus (N=434x383, L=20)		Cones (N=450x375, L=60)	
	[%]	CPU	GPU	CPU	GPU
H4+SBP	100	11873	299	42672	1001
H4+AOM	44	5570	126	16656	411
SBP	94	37405	822	130813	2799
AOM	38	17249	340	51128	1159

AOM is not good as described in the previous section, we can say that the best algorithm is H4+AOM (GPU).

V. CONCLUSION

In this paper, an approximate belief propagation algorithm is proposed that reduces the computational time and the required amount of memory by replacing outgoing messages from a node with the averaged outgoing message. To avoid loss of accuracy caused by approximation, averaged outgoing messages are propagated from a low resolution graph to the original graph hierarchically.

The proposed method was evaluated against Middlebury stereo benchmark dataset in comparison with the standard belief propagation algorithm, and it was shown that both the computational time and the required amount of memory are drastically reduced with minor loss of accuracy. The proposed method was also evaluated on GPU and it was shown that the computational time was further reduced.

ACKNOWLEDGMENT

This study was supported in part by Program for Improvement of Research Environment for Young Researchers from Special Coordination Funds for Promoting Science and Technology (SCF) commissioned by the MEXT of Japan, and in part by Grant-in-Aid for Young Scientists (B)(21700224).

REFERENCES

[1] 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, pp. 7–42, 2002.

[2] X. Liu, O. Veksler, and J. Samarabandu, "Graph cut with ordering constraints on labels and its applications," in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition: CVPR*, 2008, pp. 1–8.

[3] S. E. Shimony, "Finding the maps for belief networks is np-hard," *Articial Intelligence*, vol. 68, pp. 399–410, 1994.

[4] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient belief propagation for early vision," *International Journal of Computer Vision: IJCV*, vol. 70, pp. 41–54, 2006.

[5] 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, 2001.

[6] D. Schlesinger and B. Flach, "Transforming an arbitrary minsum problem into a binary one," Dresden University of Technology, Tech. Rep., 2006, tUD-FI06-01.

[7] V. Kolmogorov and C. Rother, "Minimizing non-submodular functions with graph cuts - a review," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 29, no. 7, pp. 1274–1279, 2007.

[8] W. T. Freeman, E. C. Pasztor, and O. T. Carmichael, "Learning low-level vision," *International Journal of Computer Vision: IJCV*, vol. 40, pp. 25–47, 2000.

[9] Q. Yang, L. Wang, R. Yang, S. Wang, M. Liao, and D. Nister, "Real-time global stereo matching using hierarchical belief propagation," in *Proc. of BMVC*, 2006, pp. 989–998.

[10] T. Yu, R.-S. Lin, B. Super, and B. Tang, "Efficient message representations for belief propagation," in *Proc. of ICCV*, 2007, pp. 1–8.

[11] C.-K. Liang, C.-C. Cheng, Y.-C. Lai, L.-G. Chen, and H. H. Chen, "Hardware-efficient belief propagation," in *Proc. of CVPR*, 2009, pp. 80–87.

[12] Q. Yang, L. Wang, and N. Ahuja, "A constant-space belief propagation algorithm for stereo matching," in *Proc. of CVPR*, 2010, pp. 1–8.

[13] "http://vision.middlebury.edu/stereo/."

[14] *NVIDIA CUDA Programming Guide*, NVIDIA Corporation, 2009, version 2.3.