ANALYSIS OF BELIEF PROPAGATION FOR HARDWARE REALIZATION

Chao-Chung Cheng[1], *Chia-Kai Liang*[2], *Yen-Chieh Lai*[3], *Homer H. Chen*[2], *and Liang-Gee Chen*[1]

Graduate Institute of Electronics Engineering[1],
Graduate Institute of Communication Engineering[2],
Department of Electrical Engineering[3],
National Taiwan University

ABSTRACT

Belief propagation has become a popular technique for solving computer vision problems, such as stereo estimation and image denoising. However, it requires large memory and bandwidth, and hence naïve hardware implementation is prohibitive. In this paper, we first analyze the memory and bandwidth requirements of the technique from the hardware perspective. Then, we propose a tile-based belief propagation algorithm that works with existing data reuse schemes and achieves bandwidth reduction by a factor of 10 to 400. We apply the proposed algorithm to stereo estimation and show that its performance is comparable to the original algorithm.

Index Terms— Belief propagation, hardware implementation, stereo estimation.

1. INTRODUCTION

Many problems in computer vision and image processing attempt to assign an optimal label to each node (pixel, block, or some other element) of a scene representation. A label stands for a local quantity. For example, the label for a pixel is the disparity vector in stereo estimation and the motion vector in motion/optical flow estimation.

Finding the optimal label assignment can be formulated as a problem of energy (cost) minimization on a Markov Random Field (MRF). The energy has two terms: a data term E_d that penalizes the inconsistency between the labels and the observed data, and a smoothness term E_s that favors the spatial coherence of the labels. The optimal labels $\{l_p\}$ are the labels that minimize the combination of these two terms,

$$\{l_p\} = \arg\min\left\{\sum_{p\in P} E_d(l_p) + \sum_{(p,q)\in G} E_s(l_p, l_q)\right\}, \tag{1}$$

where P is the set of all nodes and G is a specified neighborhood, such as the 4-nearest neighboring pixels.

Though the MRF formulation was proposed more than 20 years ago [1], the problem is NP-hard. In the late nineties, efficient algorithms such as graph cuts [2], (loopy) belief propagation [3], and numerous variants were proposed [4]. These algorithms can find strong locally optimal solution in polynomial time and enable many applications such as image denoising, inpainting, image stitching, bi-layer segmentation, etc [4].

While the software implementation of these algorithms is generally free of resource constraint, the hardware implementation needs to consider bandwidth, internal and external memory size, degree of parallelism, regularity of memory access, etc. Therefore, an algorithm efficient on software may not be suitable for hardware implementation. This is particularly true for portable devices, such as cell phones and digital cameras.

In our initial study, we find that belief propagation has high potential for hardware implementation. It is highly parallel and only uses simple operations. However, it requires huge memory and bandwidth. Thus, the straightforward hardware implementation is prohibitive.

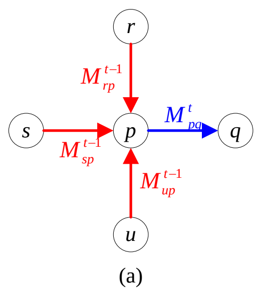
In this paper, we first focus on the bandwidth and memory analysis of the belief propagation technique. Then we propose a tile-based belief propagation algorithm to overcome the memory and bandwidth bottlenecks. Compared to the original belief propagation, the proposed algorithm has similar performance with much lower bandwidth and memory requirements. Therefore, it is more suitable for hardware implementation than the original algorithm.

The rest of the paper is organized as follows. In Section 2, we review the original belief propagation and analyze its bandwidth and memory consumption. In Section 3, we present the proposed tile-based belief propagation in detail. In Section 4, we test the proposed algorithm on stereo estimation. We conclude this paper and discuss future research directions in Section 5.

2. BELIEF PROPAGATION

The belief propagation (BP) iteratively performs the message passing operations. At iteration t, each node p sends a $|L|$-dimensional message M_{pq}^t to its neighbor q. Each entity $M_{pq}^t(l)$ in the message is

$$M_{pq}^t(l_q) = \min_{l'\in L}\left\{E_s(l_q, l') + E_d(l') + \sum_{(p,p')\in \mathbf{N}_p\setminus q} M_{p'p}^{t-1}(l')\right\}, \tag{2}$$



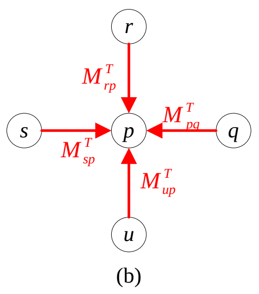


Figure 1. (a) A message at iteration t from p to q is constructed using the messages from r, s, and u to p at iteration $t-1$. (b) The node p collects all messages from the neighbors to decide the best label.

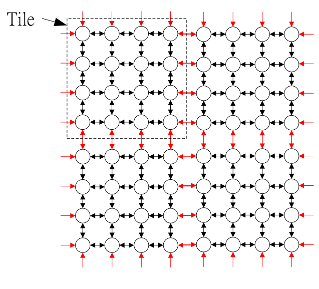
where L is the set of all labels, $|L|$ is the number of labels, and $\mathbf{N}_p$ is the set of the neighbors of p (Fig. 1(a)). $M^t_{pq}(l)$ encodes the opinion of p about assigning label l_q to q. Node p first scans all labels l' and decides the one having the greatest support for assigning l to q based on 1) the smoothness (compatibility) cost between l' and l (the first term in Eq. (2)), 2) the self-judgment of p about being assigned l' (the second term), and 3) The opinion from neighbors expect q ($\mathbf{N}_p\backslash q$) about assigning l' (the third term). During BP, all nodes exchange messages (opinions) about the label assignment and through iterations, nodes far away from p can influence p's label assignment.

The messages are iteratively propagated. As the BP-M method in [4], we define a single iteration as propagating a message from the top-left node to the bottom right one, and then propagating a message from the bottom-left node to the top-left one. After enough number of iterations, say T, the label of p is determined based on the local likelihood and the messages from the neighbors (Fig. 1(b)):

$$l_p = \arg\min_{l\in L}\left\{E_d(l) + \sum_{(p,p')\in N(p)} M^T_{p'p}(l)\right\}. \qquad (3)$$

To this point we can clearly see the advantages of BP for hardware implementation. Firstly, it is highly parallel. In message passing, each node loads the messages from the previous iteration, operates independently, and generates new messages. Secondly, it only uses simple operations such as additions and comparisons. Third, the memory access is regular. If we update the message sequentially, the required input data can be streamed into the processor with ease. On the contrary, other algorithms such as graph cut require complicated operations like tree construction and sorting. They also frequently perform the random memory access, which obstruct the efficient hardware pipeline.

However, BP cannot be efficiently implemented in hardware due to the huge memory and bandwidth consumption. Like other MRF algorithms, it stores $N|L|$ data terms, where N is the number of nodes (the smoothness term can usually be analytically calculated on-the-fly). BP stores extra $4N|L|$ messages for the 4-connected neighborhood system. Each neighboring pair has two messages, one for each direction. Therefore, it totally needs to store $5N|L|$ elements. For example, in stereo estimation on a VGA image pair with the disparity range of 16, assume each message and data term takes 1 bytes, BP totally takes 24,576,000 bytes, 80 times the size of the image.



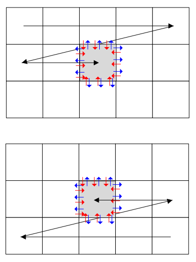


Figure 2. (a) For a small block (black nodes), the messages around it (red arrows) give enough information about the outside world. (b) The proposed tile-based belief propagation method. The tiles are first processed in a raster scan order and then in an inverse-raster scan order.

In performing message passing, each node loads 3 messages and $|L|$ data term and outputs 1 new message. Because for each node there are four outgoing messages, BP requires $20N|L|$ data transferring per iteration and $20N|L|T$ for convergence. Using the example above and assume BP converges at T=50, this amounts to 4,915,200,000 (4.58G) bytes data transferring. For real-time video application, it corresponds to 137.33GB per second. Obviously this daunting bandwidth requirement is infeasible for the existing hardware, as discussed in [8].

Therefore, proper data reuse and data reduction strategies must be applied for efficient hardware design. In [6], several compression techniques are proposed to compress the messages. However, the number of data transfer is unchanged. Another trivial method is to partition the image into many blocks and perform BP within each block independently [7]. However, this approach downgrades the belief propagation from a global technique into a local one and results in bad local minimums (See Table 2 and Fig. 6 and 7). In the next section, we attack the bandwidth bottleneck at the algorithm level by using a better message passing method. However, the proposed method can cooperate with many existing data reuse method.

3. TILE-BASED BELIEF PROPAGATION

Before describing the proposed algorithm, we first look at the message passing procedure *locally* at a small tile (Fig. 2(a)). We can see that for this small region, we do not have to care about how messages outside the region (red ones in Fig. 2(a)) are constructed. As long as the procedure is toward convergence, these messages should carry the correct opinions about the labeling of this region from other regions. According to the Markovian property, knowing these messages is identical to knowing all information of the nodes outside the current tile.

This assumption is verified by a simple experiment. We setup a MRF with simple energy definition and perform BP. After convergence, we reset the messages within a specific region and re-run BP with this region by fixing the message around the boundary of the region. After convergence again, we find that the new messages are almost identical to the original ones before reset. In other words, given the boundary messages and the data terms and smoothness terms of a local patch, the messages inside can be thrown away without losing any information.

3.1. Proposed Algorithm

According to this observation, we propose a tile-based belief propagation method. The algorithm is illustrated in Fig. 2(b) and the pseudocode is shown in Fig. 3. In the beginning, the image is split into non-overlapping tiles of size $B{\times}B$. The algorithm has a two-level structure: outer and inner iterations. In the outer iteration, the tiles are first processed in a raster scan order (line 3). For each tile, the messages from other tiles (red ones in Fig. 2 (b)) and the date terms E_s are loaded and then BP (Eq. (2)) is preformed within the tile (BPinOneTile in Fig. 3). After T_i inner iterations (line 6-7), the messages sending outside the tile are stored (blue ones in Fig. 2(b)). The inner iterations acts like a *filtering* operation that purifies the messages according to the data terms and smoothness terms within the tile. At the end of the scan, we reach the bottom-right tile. We then perform the same procedure in an inverse-raster scan order (line 9-17, Fig. 2(b)). At the T_o outer iteration, the best labels are determined using Eq. (3).

3.2. Bandwidth and Memory Analysis

3.2.1 Proposed Tile-based Belief Propagation

The main advantage of the proposed algorithm is that compared with the original BP, the memory and bandwidth consumption is greatly reduced. We do not need to store the messages inside the tile. In each new iteration, the messages inside can be re-generated from the boundary messages and data terms. Therefore, the off-chip storage of the messages becomes:

$$(W/B)\cdot(H/B)\cdot 4B|L| = 4N|L|/B \tag{4}$$

where W and H are the width and height of the image, respectively (WH=N). The reduction is a factor of B. Similarly, the bandwidth becomes:

$$2(\underbrace{N|L|}_{\text{Data terms}}+\underbrace{4N|L|/B}_{\text{Incoming messages}}+\underbrace{4N|L|/B}_{\text{Outgoing messages}})T_o=(2N|L|+16N|L|/B)T_o. \tag{5}$$

The factor 2 is because each tile is processed twice at each iteration.

```
function {$l_p$}←TileBasedBP($E_d$, $E_s$, $B$, $T_o$, $T_i$)
1   Initialize all message entities $M_{pq}^0(l)$=0
2   for $t_o$=1,…,$T_o$
3       loop through all tiles in a raster scan order
4           Load $M_{pq}^{t-1}$ for $p\notin C$ and $q\in C$; //$C$ is the current tile.
5           Load $E_d(l_p)$ for $p\in C$;
6           for $t_i$=1,…,$T_i$
7               {$M_{pq}^t$} ← BPinOneTile({$M_{pq}^{t-1}$}, $E_d$, $C$);
8           Store {$M_{pq}^t$}for $p\in C$ and $q\notin C$;
9       loop through all tiles in a inverse-raster scan order
10          Load {$M_{pq}^{t-1}$} for $p\notin C$ and $q\in C$;
11          Load $E_d(l_p)$ for $p\in C$;
12          for $t_i$=1,…,$T_i$
13              {$M_{pq}^t$}← BPinOneTile({$M_{pq}^{t-1}$}, $E_d$, $C$);
14          if ($t_o$ = $T_o$) Obtain {$l_p$} for $p\in C$; //using Eq. (3);
15          else Store {$M_{pq}^t$} for $p\in C$ and $q\notin C$;
16  return {$l_p$};
```

```
function {$M_{pq}^t$}← BPinOneTile({$M_{pq}^{t-1}$}, $E_d$, $C$)
17  Initialize $M_{pq}^t(l)$=0 for all $p\in C$ and $q\in C$;
18      Update rightward messages; //using Eq. (2);
19      Update leftward messages;
20      Update downward messages;
21      Update upward messages;
22  return {$M_{pq}^t$};
```

Figure 3. The pseudocode of the proposed algorithm.

We use the same example, stereo estimation on a dataset of VGA size, and the disparity range is 16, to demonstrate the memory and bandwidth reduction of the proposed algorithm. Because each algorithm has different convergence property, we assume the number of the (outer) iteration is 1 here. The comparison is shown in Table 1. For the original BP, when there is only one PE, it requires very small internal memory. However, it requires huge external memory and bandwidth. The block-based BP processes B^2 nodes at a time and thus the $4B^2$ messages between the nodes of the same block can be loaded at once. After the messages are updated, they can be sent to the external together. This method can save the bandwidth to 45%. However, the size of the external memory is unchanged since all messages must be stored. Also the reduction factor does not change with the block size. This limits the flexibility of the design. Furthermore, because the messages between the blocks are dropped, the results could be disastrous (see Section 4).

The proposed tile-based BP also performs the message passing one tile at a time. However, only the messages around the tile are stored and transferred. Therefore, the external memory reduces to 21.25~30% and the bandwidth reduces to 5.625~10%, depending on the tile size. Although the tile-based BP requires a larger internal memory than the original BP does, this cost is affordable to the current technology.

		Original BP [3]	Proposed tile-based BP				Proposed tile-based BP + data reuse			
Block(tile)-size (B^2)		–	8^2	16^2	32^2	64^2	8^2	16^2	32^2	64^2
Internal memory	Data terms	16	1,024	4,096	16,384	65,536	1,280	4,864	18,944	74,752
	Messages	64	4,096	16,384	65,536	262,144	4,096	16,384	65,536	262,144
	Total	80	5,120	20,480	81,920	327,680	5,376	21,248	84,480	336,896
	Factor (to the same tile size)		100%	100%	100%	100%	105.0%	103.8%	103.1%	102.8%
External memory	Data terms	4,915,200	4,915,200				0			
	Messages	19,660,800	2,457,600	1,228,800	614,400	307,200	2,457,600	1,228,800	614,400	307,200
	Total	24,576,000	7,372,800	6,144,000	5,529,600	5,222,400	2,457,600	1,228,800	614,400	307,200
	Factor (to original BP)	100.00%	30%	25%	22.5%	21.25%	10%	5%	2.5%	1.25%
Bandwidth	Data terms	19,660,800	4,915,200				614,400			
	Messages	78,643,200	4,915,200	2,457,600	1,228,800	614,400	4,915,200	2,457,600	1,228,800	614,400
	Total	98,304,000	9,830,400	7,372,800	6,144,000	5,529,600	5,529,600	3,072,000	1,843,200	1,228,800
	Factor (to original BP)	100%	10%	7.5%	6.25%	5.625%	5.625%	3.125%	1.875%	1.25%

Table 1. The memory and bandwidth consumption of the original belief propagation (BP) [3], the proposed tiles-based BP, and the tile-based BP plus the data reuse technique for on-line calculating the data terms (without counting the external memory size of the image pair).

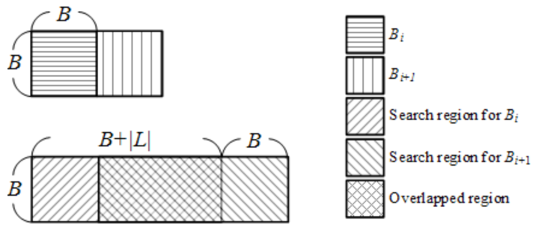


Figure 4. Level-C data reuse for calculating the data terms. B_i and B_{i-1} are two successive blocks in the left image. Because most of their search region in the right image is overlapped, besides the block B_{i+1} itself, only B^2 pixels are needed to be loaded.

3.2.2 Data Reuse for Data Term Calculation

While the proposed algorithm greatly reduces the memory and bandwidth, the advantages of the original BP are still preserved. The algorithm is still highly parallel. We can use B identical process elements (PE's) to calculate one row (column) of new messages in parallel. Furthermore, because the algorithm is tile-based, several pipeline and data reuse techniques used in the video compression can be applied [5] to lower the bandwidth by using slightly more internal memory.

For example, in stereo estimation, the data term $E_d(l_p)$ is usually defined as a function of the intensity difference between the pixel (p_x, p_y) in left image I_L and the pixel (p_x+l_p, p_y) in right image I_R:

$$E_d\left(l_p\right) = f\left(I_L\left(p_x, p_y\right) - I_R\left(p_x + l_p, p_y\right)\right). \tag{6}$$

Because the proposed algorithm is performed in tile-based, we can . When performing BP for one tile, we can load the images and construct the data terms of the next tile. Using the level-C memory reuse scheme [5], we only need to load $2B^2$ bytes for calculating the data cost of one tile, as shown in Fig. 4. Therefore, the external memory for the data terms are eliminated and the overall bandwidth decreases from $(2N|L|+16N|L|/B)T$ to:

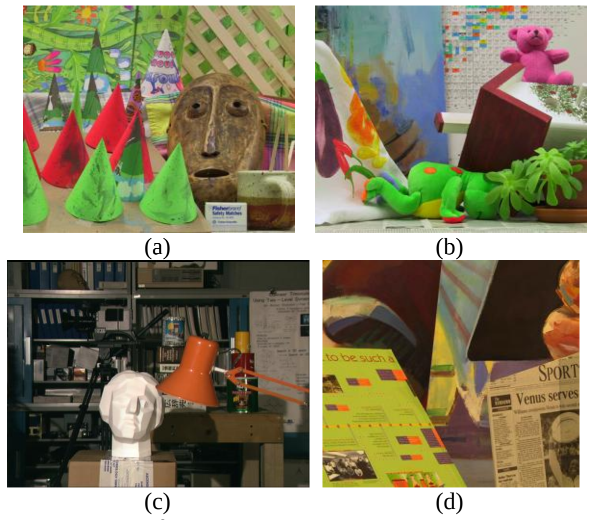


Figure 5. Dataset for stereo estimation. (a) *Cones*, (b) *Teddy*, (c) *Tsukuba*, and (d) *Venus*.

$$2(\underbrace{2N}_{\text{image pixels}} + \underbrace{8N|L|/B}_{\text{messages}})T_o = (4N + 16N|L|/B)T. \tag{7}$$

And the internal memory size increases from $|L|B^2$ to

$$\underbrace{|L|B^2}_{\text{data term}} + \underbrace{(B+L)B}_{\text{search range}} + \underbrace{B^2}_{\textit{current}\text{ tile}}. \tag{8}$$

Using this technique, we can further reduce the memory and bandwidth consumption, as shown in Table 1. Compared with to original BP, the external memory is reduced by a factor of 10~80 and the bandwidth is reduced by a factor of 18~80.

4. EXPRIMENTAL RESULTS

We have shown that the proposed algorithm can significantly reduce the memory and bandwidth. Here we use the stereo estimation to show that performance-wise, our algorithm is much better than the block-based BP and comparable to the original BP.

		Cones	*Teddy*	*Tsukuba*	*Venus*	Avg.
Original BP [3]		2466383	2227224	333466	802006	
Block-base BP [7]	B=32	2554895	2308736	351942	853686	
		3.59%	3.66%	5.54%	6.44%	4.81%
	B=64	2520843	2271426	350840	826655	
		2.21%	1.98%	5.21%	3.07%	3.12%
Tile-based BP	B = 32	2498884	2253897	333982	819649	
		1.32%	1.20%	0.15%	2.20%	1.22%
	B = 64	2486742	2241247	345227	817493	
		0.83%	0.63%	3.53%	1.93%	1.73%

Table 2. The energy of the solutions using different BP algorithms and the increase of the energy to the original BP.

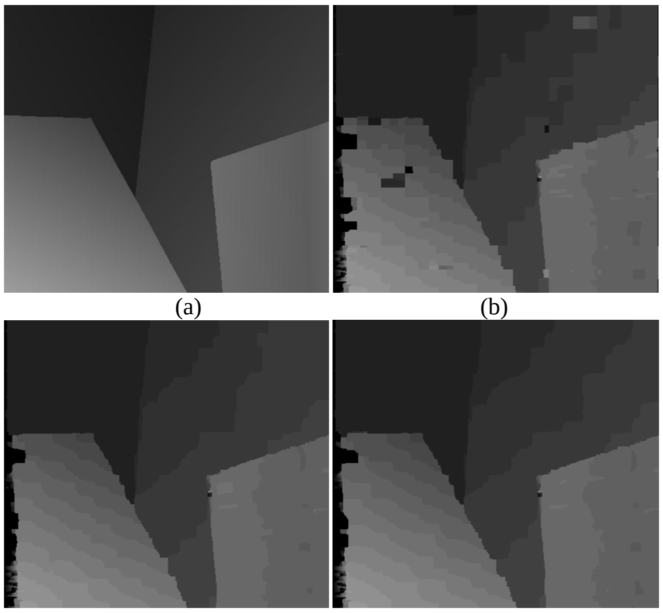


Figure 6. The disparity maps of *Venus*: (a) Ground truth, (b) the block-based BP, (c) the original BP, and (D) the tile-based BP. The block size (B) is 32.

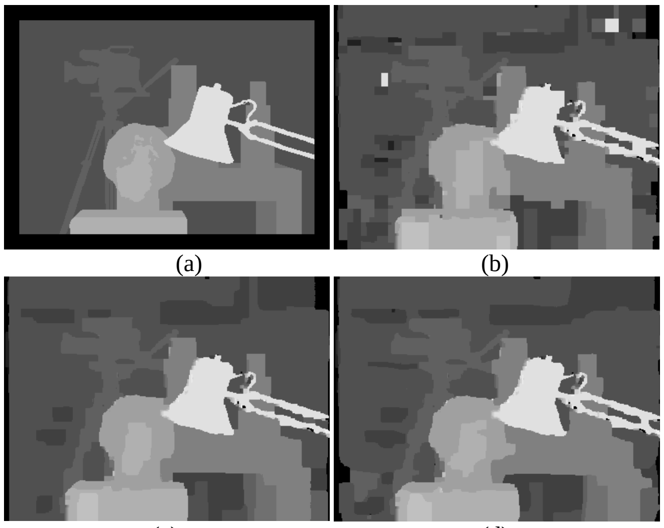


Figure 7. The disparity maps of *Tsukuba*: (a) Ground truth, (b) the block-based BP, (c) the original BP, and (d) the tile-based BP. The block size (B) is 16.

We use four dataset: *Cones*, *Teddy*, *Tsukuba*, and *Venus*, from the Middlebury vision website [9] (Fig. 5 (a-d)). These dataset are commonly used in the computer vision community for performance measurement [10]. Since our goal is to compare the performance of difference BP algorithms, we adopt the simple data and smoothness term definitions according to [2]. The data term is the Birchfield-Tomasi method which can minimize the sampling effect [11]. The smoothness term is a generalized Potts model:

$$E_s\left(l_p,l_q\right)=\lambda\min\left(\left|l_p-l_q\right|,T_s\right). \tag{9}$$

For a fair comparison, we set T_s=2 and λ=20 for all BP algorithms.

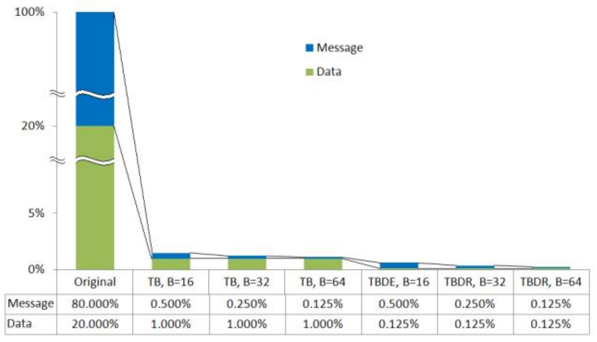
Different BP algorithms require different number of iterations. To reach the performance limitation of the algorithms, we empirically choose the number of the iterations to be large enough such that all algorithms converge for all dataset. For the original BP and the block-based BP, T is 50; for the tile-based BP, inner iteration T_i is set to 8 for B=32 and 24 for B=64, and T_o is 10 (In fact, setting T_o=3 gives visually plausible results).

We first examine the energy value of the difference algorithms after convergence, as summarized in Table 2. Compared with the block-based BP, the solution found by the tile-based BP is more close to the one found by the original BP. For example, when B = 32, block-base BP increases the energy by 3.59 to 6.44% (avg. 4.81%), and the proposed tile-based BP only increases the energy by 0.15 to 2.20% (avg. 1.22%). Actually, in all our experiments, after the first outer iteration, our energies are already below the ones obtained by the block-based BP after 50 iterations.

Not only the energy obtained by the tile-based BP is closer to that obtained by the original BP, the resulting disparity is also more accurate. The estimated disparity maps of *Venus* and *Tsukuba* are shown in Fig. 6 and 7, respectively. We can see that the block-based BP results in serious blocky artifacts because the messages between the neighboring blocks are dropped. On the other hand, the disparity generated by our method is very similar to the one generated by the original BP.

Since the number of the inner iterations does not affect the bandwidth, the tile-based BP can reduce the bandwidth cost by using a smaller number of iteration. This is illustrated in Fig. 8 in which the iteration number is considered now (note that it was assumed to be one in the previous section). We can see that our algorithm only requires 1.125~1.5% bandwidth of the original BP. Using data reuse, the bandwidth can be further reduced to 0.25%, only 1/400 of the original requirement.

Although we only show the application of the simple stereo estimation, our tile-based BP can be used to other applications since the overall performance is similar to the original BP. The simple stereo estimation used in our expe-



	Original	TB, B=16	TB, B=32	TB, B=64	TBDE, B=16	TBDR, B=32	TBDR, B=64
Message	80.000%	0.500%	0.250%	0.125%	0.500%	0.250%	0.125%
Data	20.000%	1.000%	1.000%	1.000%	0.125%	0.125%	0.125%

Figure 8. The bandwidth cost of the original BP, the tile-based (TB) BP, and the TB BP with data reuse (TBDR) of different block sizes (*B* = 16, 32, and 64).

riment is not comparable to the start-of-the-art algorithms, but it is due to the usage of the simple energy functions. In fact, all the top seven stereo estimation algorithms on the Middlebury website choose the belief propagation to minimize the complex energy functions they define. Therefore, our tile-based BP is a good candidate of hardware implementation of these algorithms.

5. CONCLUSION

Belief propagation is a global energy minimization technique. Although it gives a better solution than the local minimization methods, it is more difficult for hardware implementation. In this paper, we have analyzed the memory and bandwidth requirements of the original BP algorithm and showed that the bottleneck is resulted from the huge number of the messages to be stored and transferred.

By characterizing the property of the messages, we have developed a tile-based propagation algorithm. Since it only stores and passes the boundary messages of the tiles, a great reduction of memory and bandwidth is achieved. Also, it converges to a solution of quality similar to that generated by the original algorithm. Therefore, the proposed algorithm is more suitable for hardware implementation than the original algorithm.

There are several topics to be investigated as future work. First, because the message passing scheme differs from that of the original BP, a good balance between the inner and outer iterations is required. If we use a large number of inner iterations for a tile, the resulting outgoing messages would only represent the local opinions of this tile, while the information coming from far regions are filtered out. In other words, these messages can be easily stuck in a local optimal state.

Second, existing MRF applications often use more complex data terms and smoothness terms than the ones we used in our experiments. How to calculate and store these data efficiently is an important topic. The data reuse scheme used here needs to be refined.

Finally, the complexity of the proposed tile-based BP needs to be addressed. Our initial study suggests that its arithmetic complexity is similar to that of the original BP. However, tile-based BP requires a smaller number of iterations and thus the actual computational complexity should be lower. It should also be investigated how an efficient BP software such as the one described in [12] can be ported onto hardware.

7. ACKNOWLEDGEMENT

Part of this project is supported by Himax Technologies, Inc.

8. REFERENCES

[1] S. Geman and D. Geman, “Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images,” *IEEE Trans PAMI*, vol. 6, no. 6, pp. 721-741, 1984.

[2] Y. Boykov, O. Veksler, and R. Zabih, “Fast Approximate Energy Minimization via Graph Cuts,” in *Proc. ICCV*, vol. 1 pp. 377-384, 1999.

[3] W. Freeman, E. C. Pasztor, and O. T. Carmichael, “Learning the Low Level Vision,” *IJCV*, vol. 70, no. 1, pp. 41-54, 2000.

[4] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother, “A Comparative Study of Energy Minimization Methods for Markov Random Fields with Smoothness-Based Priors,” *IEEE Trans PAMI*, vol. 30, no. 6, pp. 1068-1080, 2008.

[5] J.-C. Tuan, T.-S. Chang, and C.-W. Jen, “On the Data Reuse and Memory Bandwidth Analysis for Full-Search Block-Matching VLSI architecture,” *IEEE Trans CSVT*, vol. 12, no. 1, pp. 61-72, 2002.

[6] T. Yu, R.-S. Lin, B. Super, and B. Tang, “Efficient Message Representations for Belief Propagation,” in *Proc. ICCV*, 2007.

[7] Y-.C. Yseng, N. Chang, and T.-S. Chang, “Low Memory Cost Block-Based Belief Propagation for Stereo Correspondence,” in *Proc. ICME*, pp. 1415-1418, 2007.

[8] T.-C. Chen, S.-Y. Chien, Y.-W. Huang, C.-H. Tsai, C.-Y. Chen, T.-W. Chen, and L.-G. Chen, “Analysis and Architecture Design of an HDTV720p 30 Frames/s H.264/AVC Encoder,” *IEEE Trans CSVT*, vol. 16, no. 6, pp. 673-688, 2006.

[9] The Middlebury computer vision pages. Available: http://vision.middlebury.edu/

[10] D. Scharstein and R. Szeliski, “A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms”, *IJCV*, vol. 47, pp.7-42, 2002.

[11] S. Birchfield and C. Tomasi, “A Pixel Dissimilarity that is Insensitive to Image Sampling,” *IEEE Trans PAMI*, vol. 20, no. 4, pp. 401-406, 1998.

[12] P. F. Felzenszwalb and D. P. Huttenlocher, “Efficient Belief Propagation for Early Vision,” *IJCV*, vol. 70, no. 1, pp. 41-54, 2006.