Architecture Design of Belief Propagation for Real-Time Disparity Estimation

Yu-Cheng Tseng, *Student Member, IEEE,* and Tian-Sheuan Chang, *Senior Member, IEEE*

***Abstract*—Belief propagation based algorithms perform best in disparity estimation but suffer from high computational complexity and storage, especially in message passing. This paper proposes an efficient architecture design with three techniques to solve the problems. For the memory storage, we propose the spinning-message and the sliding-bipartite node plane that can reduce memory cost to 1.2% for image-scale algorithms and 23.4% for block-scale algorithms, when compared to the traditional approach. For the logic complexity, we propose a buffer-free processing element architecture that has 3.6 times hardware efficiency of the previous work. The three proposed techniques could be applied to various belief propagation based algorithms to save significant hardware cost as well as approach real-time speed.**

***Index Terms*—Belief propagation, disparity estimation.**

I. Introduction

WITH EMERGING stereoscopic displays [1]–[3], stereoscopic videos have become popular recently. In contrast with 2-D videos, stereoscopic videos demand depth information of scenes. The depth information can be derived from disparity maps, which describe the distance of correspondences in a stereo pair of images. The disparity maps can be used to synthesize virtual views for free-viewpoint videos [4] or stereoscopic displays, as well as predict inter-view motion for video coding, such as multi-view video coding [5] and free-viewpoint television [6]. For these applications, the disparity estimation design requires real-time processing speed and high quality results.

Fig. 1 shows the concept of disparity estimation. In Fig. 1(a), the two cameras C_L and C_R capture the object point and project it at P_L and P_R on the epipolar line. Given the focus f and the baseline B of the cameras, if we could estimate the disparity $X_R - X_L$, the object depth Z can be acquired by $fB/(X_R - X_L)$. Thus, disparity estimation algorithms attempt to find all correspondences in a pair of images.

The local approaches use the pixels within a window to find correspondences. Hence, they have low computational complexity and easily achieve real-time processing by the FPGA [9] and GPU software-based [10] implementation. However, they suffer from inaccurate disparity. On the other hand, the global approaches further apply the disparity optimization, such as dynamic programming (DP) [11], graph-cut (GC) [12], and belief propagation (BP) [13]. Because they consider disparity selection in a whole image, they can deliver better disparity maps but suffer from high computational complexity. In these approaches, the results by the DP have the shrinking artifact due to its 1-D scan line process. To solve the shrinking artifact, Gong [14] and Park [15] proposed some additional schemes and implemented them to achieve real-time processing by GPU and FPGA based designs. In contrast, the GC and the BP are performed in a 2-D space and have better accuracy than the DP-based algorithms as shown in the Middlebury evaluation [16]. The computational complexities of GC and BP are $O(L^3)$ and $O(L^2)$, where L is the disparity range. Gallup *et al*. [17] adopted the GC with L equal to 3 to refine sweeping planes for 3-D reconstruction. However, the GC is an optional process in their real-time implementation since it spends about 2 s for a frame. Compared to the computation of GC-based algorithms, that of BP-based algorithms is more regular and suitable to be accelerated by parallel hardware design. Thus, this paper focuses on the state-of-art method, the BP-based algorithm.

The concept of the BP-based algorithm is illustrated in Fig. 2. In the BP-based algorithms, an energy function is generally formulated as

$$E(d) = \sum_{i \in I} D(d_i) + \sum_{i \in I,\, j \in Neighbor(i)} V(d_i, d_j) \qquad (1)$$

for a 2-D graph in Fig. 2(a). In this energy function, D is the data cost for each node corresponding to each pixel, V is the smoothness cost for each edge, and d of the energy E is a selected disparity set for all nodes. The two costs can constrain selecting the disparity set d. The data cost D enforces that the correspondences are similar, and the smoothness cost V enforces that the neighboring nodes' disparities are consistent. To minimize the energy function and acquire an appropriate disparity set d, the BP-based algorithms perform an iterative process called message passing. However, the shortage of the BP-based algorithms is that the energy function may not be convergent definitely. Nevertheless, the disparity map could approach to a steady state after sufficient iterations.

For the requirement of real-time processing, the direct hardware implementation of BP-based algorithms suffers from two design challenges: high computational complexity and

Manuscript received August 27, 2009; revised January 6, 2010, May 27, 2010, and July 9, 2010; accepted August 26, 2010. Date of publication October 14, 2010; date of current version November 5, 2010. This work was sponsored by the National Science Council, Taiwan, under Grant NSC-98-2220-E-009-012. This paper was recommended by Associate Editor Y.-S. Ho.

The authors are with the Department of Electronics Engineering, National Chiao-Tung University (NCTU), Hsinchu 30010, Taiwan (e-mail: tyucheng@dragons.ee.nctu.edu.tw; tschang@dragons.ee.nctu.edu.tw).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSVT.2010.2087434

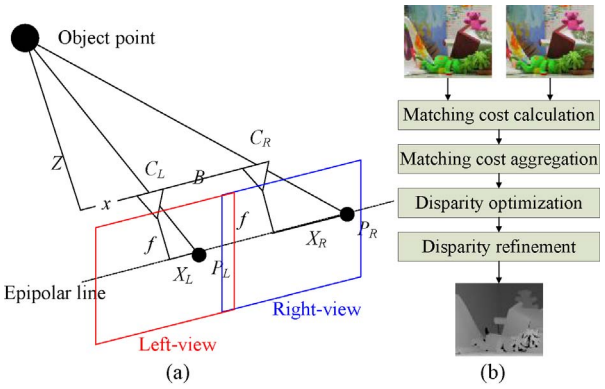


Fig. 1. (a) Geometry of disparity estimation. (b) Framework of disparity estimation.

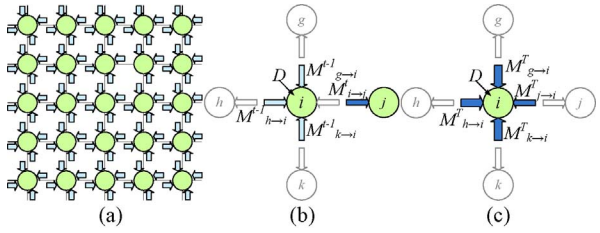


Fig. 2. Illustrations of BP. (a) Node plane. (b) Message passing. (c) Belief calculation.

storage in the message passing. For the example of 640 × 480 at 30 frames/s and the disparity range of 32, the computational complexity is about 1200 billion operations per second for the message passing, and the storage is about 157 MB for messages.

To address above problems, various approaches have been proposed. Felzenszwalb and Huttenlocher [18] proposed an efficient message passing to reduce computational complexity from $O(L^2)$ to $O(L)$, and the bipartite message approach to reduce 50% memory cost. Following their approach, Yang *et al*. [19] implemented it on a high performance GPU, and Park *et al*. [20] also designed an array processing architecture on two FPGA boards to achieve the performance of 320 × 240 at 30 frames/s but with 880 kB on-chip memory. Cheng *et al*. [22]–[24] proposed a tile-based BP and a fully parallel architecture for each message passing processing element (PE) to reach real-time processing for the image size of 640 × 480. Nevertheless, all the implementation still suffers from high memory cost. In summary, though previous work used parallel PEs to conquer the high complexity, the resulted logic still occupies too much area since each PE needs high area cost. In addition, all the work did not solve the memory cost well due to their fixed memory access approach.

To solve the mentioned problems, we propose a hardware efficient architecture for various BP-based algorithms through three techniques. For the high memory cost, we propose a spinning-message approach which rearranges the message configuration in an internal memory to save 50% memory cost. In addition, we propose a sliding-bipartite node plane that combines the advantages of previous work to further reduce more memory cost. For the message passing PE, we propose a buffer-free PE architecture which removes all the large buffers and shares common operators to reduce logic cost without significant speed degradation. Both the proposed low memory access approaches and the buffer-free PE architecture could be applied to various BP-based algorithms together to significantly reduce their hardware cost as well as speed up to real-time processing without changing their disparity accuracies

The rest of this paper is organized as follows. Section II reviews the BP-based algorithms and points out their design challenges. Section III presents the proposed low memory access approaches, and Section IV elaborates the buffer-free PE architecture. Then, Section V shows the implementation results and comparisons. Finally, Section VI concludes this paper.

II. Review of Belief Propagation

In this section, we review various BP-based algorithms and then indicate the general design problems in these algorithms.

A. Baseline BP

Sun *et al*. [13] first applied BP to disparity estimation. This baseline BP includes three steps: data cost calculation, message passing, and disparity selection, which are performed in the graph of Fig. 2(a). In this paper, the graph is called as node plane whose size equals to an image in the baseline BP.

In the baseline BP, the first step is to calculate the data cost of each node, where the data cost is identical to the matching cost in local approaches. According to the data cost, local approaches can determine disparity maps using the winner-take-all scheme. In contrast, the baseline BP further propagates it to neighboring nodes.

In the second step, the messages, which are the arrows in Fig. 2(a), are added around all nodes, and they propagate data cost to neighboring nodes. In the baseline BP, the propagating mechanism is called as message passing. Fig. 2(b) illustrates the message passing for calculating a new message, and its equation is as follows:

$$M^t_{i\phi j}(d_j)=\frac{1}{\kappa}\min_{d_i}\left(V(d_j,d_i)+D(d_i)+\sum_{x\in Neighboring(i)\backslash j}M^{t-1}_{x\to i}(d_i)\right) \quad (2)$$

where $M^t{}_{i\to j}$ is a new message of the node j at the iteration t from the node i, and $M^{t-1}{}_{x\to j}$ is an old message of the node i at the iteration $t-1$ from the nodes x which can be g, h, and k. In addition, V and D are smoothness cost and data cost in (1), and κ is a normalization term. Note that the indexes d_i and d_j are, respectively, for the nodes i and j. To calculate the new message $M^t{}_{i\to j}$, the three old messages $M^{t-1}{}_{h\to j}$, $M^{t-1}{}_{h\to j}$ and $M^{t-1}{}_{k\to j}$ are summed up with D by the index d_i. Then the result is convoluted with V by the cross indexes d_j and d_i. For the message passing in BP-based algorithms, the computation of (2) is performed on all four incoming messages of each node iteratively.

In the third step, the final incoming messages of each node are accumulated with its D to form a belief. The belief is

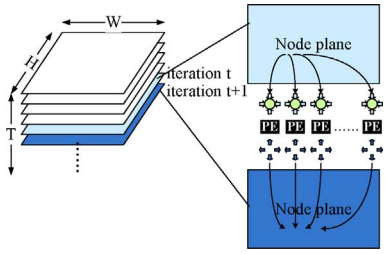


Fig. 3. Configuration of the message passing PEs.

used to determine a disparity by the following equation, and its illustration is shown in Fig. 2(c)

$$d = \arg\min \left(D(d_i) + \sum_{x \in Neighbor(i)} M_{x \to i}^{T}(d_i) \right). \quad (3)$$

In summary, the baseline BP alternates the initial data cost with the belief deriving from the message passing to deliver better disparity maps.

The major computational complexity of the baseline BP is in the message passing, and that is $O(4HWL^2T)$, where H and W are the height and width of the node plane, L is the disparity range, and T is the iteration count. The computation of the message passing can be undertaken by parallel PEs as shown in Fig. 3. These PEs use the nodes' data at the previous iteration to calculate new messages for the next iteration. With sufficient parallel PEs, the baseline BP could achieve real-time speed. However, that will result in high logic cost. In addition, high memory cost is also incurred since all the messages in the node plane have to be stored.

B. Various BP-Based Algorithms

Based on the baseline BP, various BP-based algorithms have been developed recently to address the mentioned problems from the algorithm level.

To reduce the computational complexity, Felzenszwalb and Huttenlocher [18] proposed the hierarchical BP that down-samples the node plane to multiple resolutions and then performs the message passing from coarser levels to finer levels. Because the messages in the coarser levels could propagate data cost to farther nodes and become initial messages for the next level, the disparity maps could converge faster than the baseline BP. Therefore, the hierarchical BP could take less time and deliver better disparity maps than the baseline BP.

To reduce the memory cost, our previously proposed block-based BP [25] partitions the node plane into independent blocks. The memory cost is significantly reduced from image-scale to block-scale, so that all data in the message passing can be placed in an internal memory, instead of an external memory. However, its disparity maps would suffer from blocky artifact. Furthermore, Cheng *et al*. [22] proposed the tile-based BP to improve the blocky artifact. In contrast with the independent blocks, the tile-based BP preserves the boundary messages of each tile in an external memory to link blocks.

For all the above algorithms, their computation shares the same feature: the message passing performed in a rectangular node plane. For example, the node plane is image-scale in the baseline BP and the hierarchical BP, and block-scale in the block-based BP and the tile-based BP. Therefore, in the following we will show how to develop techniques for a rectangular node plane that can be applied to various BP-based algorithms.

III. Proposed Low Memory Access Approach

In a rectangular node plane, the memory cost is constituted of the messages and the data cost. In this paper, we focus on the messages, which occupy the most of the cost. A straightforward memory access approach for the messages is the ping-pong buffer approach, which needs a pair of node planes and requires *8HWL* memory. Unfortunately, this cost is too large to be on-chip. Even if the messages are stored in an external memory, its required bandwidth is still impractical, especially for the image-scale node planes.

A. Previous Work

To reduce the memory cost of messages, Yu *et al*. [26] compressed the messages by the envelope point transform method that can achieve eight times compression without significant degradation of disparity maps. However, this compression method needs the overheads of compression and decompression.

On the other hand, much previous work focuses on the computing order of message passing on the node plane to resize the node plane for memory cost reduction. Park *et al*. [20] proposed the fast BP structure approach which resizes the pair of node planes from *HW* to *TW*, where *T* is usually smaller than *H*. In our previous work [28], we proposed the in-place message update approach that resizes one of the pair node planes from *HW* to *3W* for buffering partial new messages temporally. Felzenszwalb and Huttenlocher [18] delivered the bipartite scan which only needs one node plane, and can also reduce computation to half. Different from above computing orders, Szeliski *et al*. [21] proposed the BP-M scan which updates messages direction by direction for whole node plane to accelerate convergence speed, and only needs one node plane. Although the BP-M scan can converge faster than others, the memory cost of BP-M scan is still too high and could not be further reduced because of its iterative directional process and overlapping data lifetime in all messages. Thus, the BP-M scan is not discussed in this paper.

Excluding the BP-M scan, the memory access in the previous work belongs to the fixed memory access approach which binds messages at fixed memory positions, and thus would limit the possibility to reduce memory cost. Fig. 4 shows the data dependency of the traditional fixed memory access approach between successive iterations in a simplified 1-D node line, where each square represents a memory position, the arrow inside the square represents a stored message, and the cross line linking two messages (e.g., *m3* at *t1* to *m2* at *t2*) represents that they have data dependency. In the traditional approach, each node's messages are stored at fixed memory positions. For example, the node *n3*'s messages *m3* are always located at the same memory position *pos3* in all iterations. These messages *m3* are used to calculate the

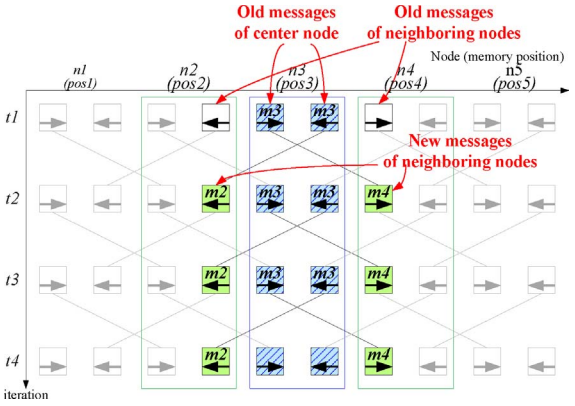


Fig. 4. Traditional fixed memory access approach in a 1-D node line for node *n3* computation.

neighboring nodes *n2* and *n4*'s new messages *m2* and *m4* for next iterations. However, the new messages cannot overwrite their old ones at the memory position *pos2* and *pos4* since their old ones are still needed for new messages computation at other nodes. Thus, an access conflict would occur between the old and new messages of the neighboring nodes. To solve the access conflict, a straightforward method is to allocate an additional memory to buffer the new messages, but it will increase extra cost.

B. Spinning-Message Approach

To address the access conflict and reduce memory cost, we propose the spinning-message approach that frees the bind between the messages and the memory positions, and eliminates the extra memory. In addition, the proposed approach could be applied to the reduction techniques mentioned in Section III-A to further save 50% memory cost.

Fig. 5(a) shows the main idea of the proposed approach. The old messages of the center node are used to calculate the new messages of the neighboring nodes, and their data life time is ended. Therefore, the new messages of the neighboring nodes can overwrite the outdated messages without access conflict, and are stored at the center memory positions instead of the neighboring memory positions.

Based on the main idea, Fig. 5(b) shows the details of the proposed spinning-message approach by a 1-D node line for the node *n3* as an example. Other nodes follow the same procedure. At the iteration *t1*, the messages *m3* are stored at the center memory position *pos3* that is the centralized mode. For the transition to the iteration *t2*, the messages *m3* are used to calculate the new messages *m2* and *m4* of the neighboring nodes *n2* and *n4*. The old messages *m3* can be replaced by the new messages at the center memory position *pos3* without the access conflict. After the calculation and replacement, the centralized mode changes to the distributed mode since every node's messages are distributed at its neighboring memory positions (e.g., *m3* at *pos2* and *pos4*) at the iteration *t2*. Then, the distributed messages *m3* are used to calculate the new messages *m2* and *m4*, and the distributed messages *m3* can also be replaced by the new messages without the access conflict. With another calculation and replacement, every node's

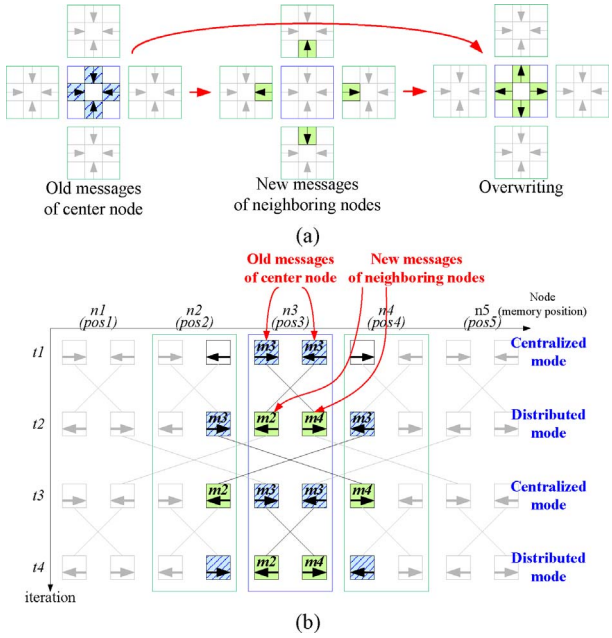


Fig. 5. Proposed spinning-message approach (a) main idea, and (b) memory access in a 1-D node line for node *n3* computation.

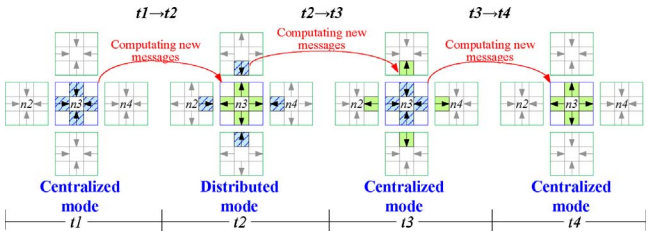


Fig. 6. Proposed spinning-message approach in a 2-D node plane for node *n3* computation.

messages are returned to the centralized mode at the iteration *t3*.

In summary, the messages are centralized at their own memory positions for odd iterations and distributed at their neighboring memory positions for even iterations. With this approach, we can save the memory while avoid the access conflict. Fig. 6 shows the proposed approach extended to a 2-D node plane.

The proposed spinning-message approach could gain two benefits. First, the memory cost could be directly reduced from *8HWL* to *4HWL* that is half of the straightforward ping-pong buffer approach. Second, the messages of each node could be updated individually, so that the computing order of message passing could be arbitrary.

C. Applications

The proposed spinning-message approach can be applied to different types of node plane to further reduce their memory cost.

1) *Sliding Node Plane:* In the original BP, the messages in a node plane are iteratively updated by the space-first (x–y plane) computing order, and the node plane moves along the iteration axis as shown in the ping-pong buffer approach of

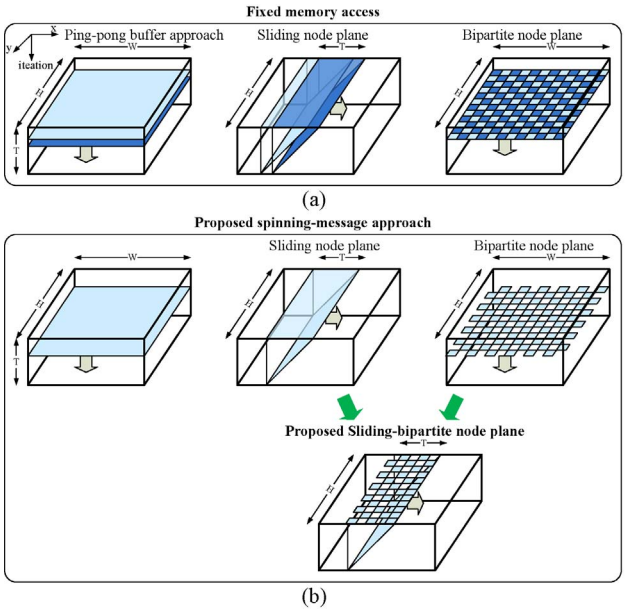


Fig. 7. Comparison of (a) the traditional fixed memory access approach and (b) the proposed spinning-message approach in different node planes.

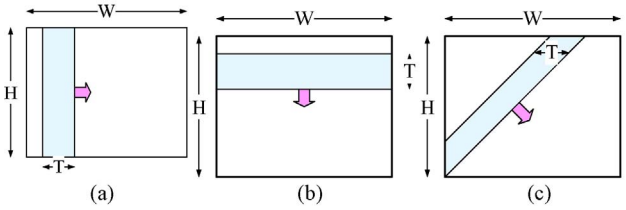


Fig. 8. Sliding node plane in different directions. (a) Vertical sliding. (b) Horizontal sliding. (c) Diagonal sliding.

Fig. 7(a). In contrast, the sliding node plane moves orthogonal to the iteration axis, and their messages are updated by the iteration-first computing order. The size of sliding node plane is its projective area on the x–y plane, which is smaller than the original node plane.

Fig. 8 shows three sliding directions. In which, the sizes of node planes are *WT* for the vertical sliding and *HT* for the horizontal sliding, and the diagonal sliding. The vertical sliding node plane was proposed by the fast BP structure approach in [20]. However, its size is larger than the other two because *W* is usually larger than *H*. Therefore, we recommend the horizontal sliding node plane, which totally requires *8HTL* memory for messages.

The memory cost can be further reduced to *4HTL* by the proposed spinning-message approach as shown in Fig. 7(b). Fig. 9 shows the details of the spinning-message approach performing on the horizontal sliding node plane. The initial state of the messages is shown in Fig. 9(a), where the front of the node plane arrives at the node *n6*. Then, in Fig. 9(b), the new messages in the node plane are computed from the node *n7* to *n2* step by step. With the spinning-message approach, the new messages can overwrite the old messages at the same memory positions. After that, in Fig. 9(c), the front of node plane will slide to the node *n7*. According to the above

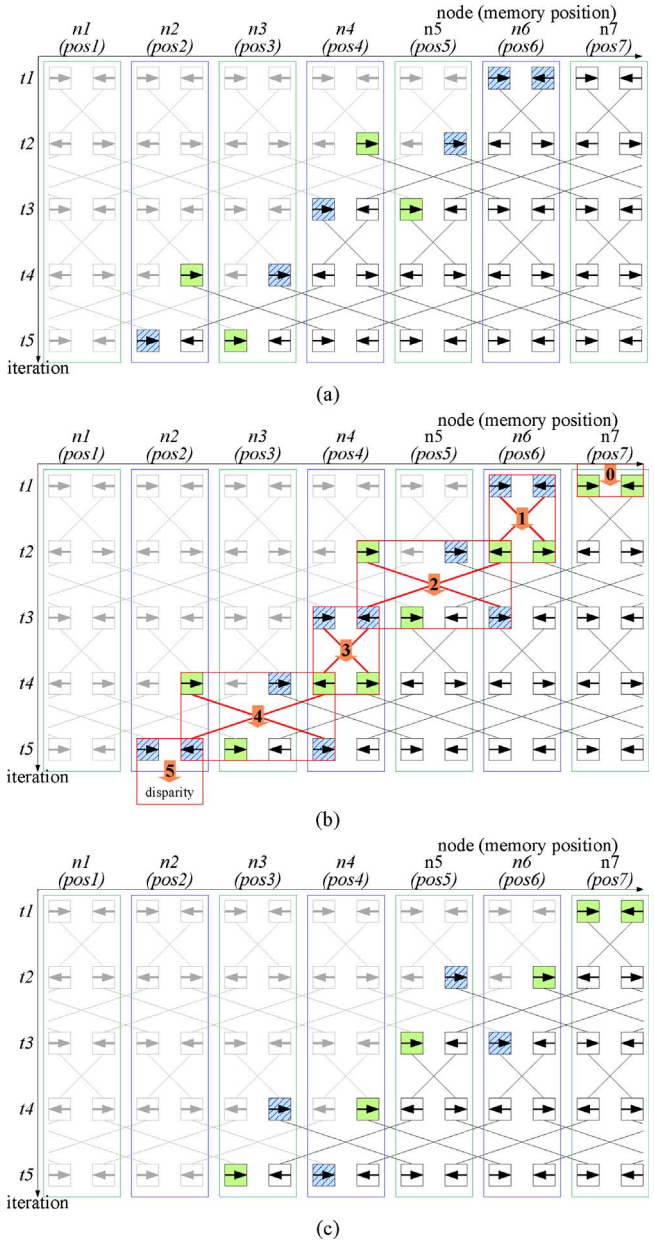


Fig. 9. Sliding node plane with the spinning-message approach. (a) Node plane slides to the node *n6*. (b) Computing order of the message passing. (c) Node plane slides to the node *n7*.

flow, the spinning-message approach could cooperate with the sliding node plane well to further save 50% memory cost.

2) *Bipartite Node Plane:* The bipartite node plane was proposed in [18] that divides nodes into two parts, like a chessboard as shown in Fig. 7(a). In which, one part is computed at odd iterations, and the other is computed at even iterations. Its memory cost is reduced from a pair of node planes in ping-pong buffer approach to only one node plane of *4HWL*.

Above memory cost can be further reduced to *2HWL* by the proposed spinning-message as shown in Fig. 7(b). Fig. 10 shows the spinning-message approach performs on the bipartite node plane at odd iterations and even iterations. At the odd iteration in Fig. 10(a), the messages of the white nodes

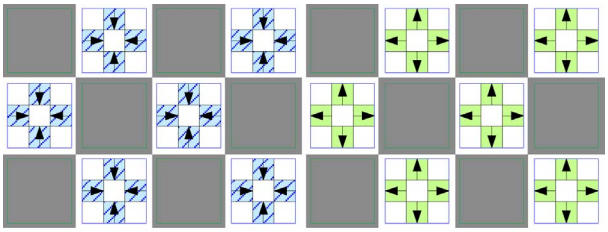


Fig. 10. Bipartite node plane with the spinning-message approach. (a) Message passing for white nodes at odd iterations. (b) Message passing for black nodes at even iterations.

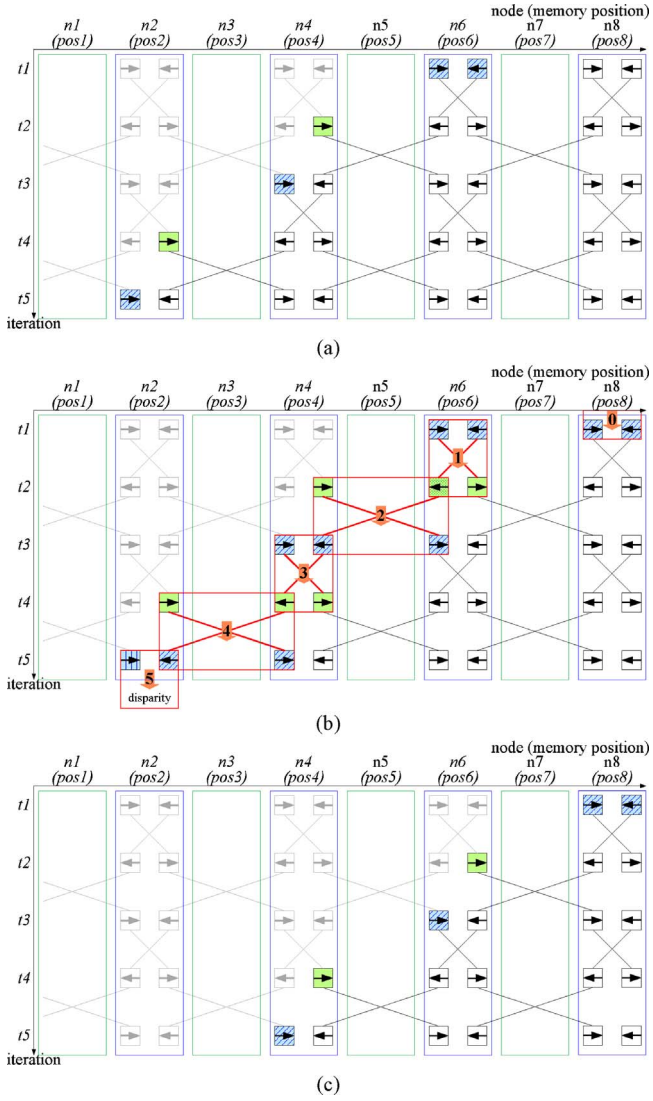


Fig. 11. Proposed sliding-bipartite node plane. (a) Node plane slides to the node *n6*. (b) Computing order of the message passing. (c) Node plane slides to the node *n8*.

are used to calculate the new messages of the black nodes, and these messages of the black nodes can overwrite those of the white nodes. Then the state of node plane is transformed to Fig. 10(b). Similarly, the messages at the even iteration can be returned to the next odd iteration. Thus by the spinning-message approach, only the white nodes need memory, and 50% memory cost can be saved.

Aggregation and forward process
1 $mf_0(-1) = MAX$
2 $mini_0 = MAX$
3 *Loop1:*
4 *for d=0 to L-1{*
5 $Ag_0(d) = D(d) + M_0^{t-1}(d) + M_2^{t-1}(d) + M_3^{t-1}(d)$
6 $mini_0 = min\{Ag_0(d), mini_0\} + Kv$
7 $mf_0(d) = min\{Ag_0(d), mf_0(d-1)\} + Cv$
8 *}*
Backward process
9 $mb_0(-1) = -MAX$
10 $norm_0 = 0$
11 *Loop2:*
12 *for d=L-1 to 0{*
13 $temp = min\{mf_0(d), temp + Cv\}$
14 $mb_0(d) = min\{temp, mini_0\}$
15 $norm_0 = norm_0 + mb_0(d)$
16 *}*
Normalization process
17 $norm_0 = norm_0 / L$
18 *Loop3:*
19 *for d=0 to L-1{*
20 $M_0'(d) = mb_0(d) - norm_0$
21 *}*

Fig. 12. Pseudo code of the message passing for calculating a new message.

3) *Proposed Sliding-Bipartite Node Plane:* By combining the above sliding node plane and bipartite node plane, the memory cost can be reduced to *4HTL*. Furthermore, applying the proposed spinning-message approach, the memory cost can be reduced to *2HTL* as shown in Fig. 7(b). Fig. 11 shows the spinning-message approach performs on the sliding-bipartite node plane. In a similar way as the sliding node plane, the front of the sliding-bipartite node plane can slide from the node *n6* to *n8* by the computing order in Fig. 11(b). Therefore, the proposed sliding-bipartite node plane takes advantages of the sliding node plane and the bipartite node plane to reduce memory cost.

IV. Proposed Buffer-Free PE Architecture

Following above proposed approaches for memory access, the message passing could be performed by parallel PEs with the configuration in Fig. 14(a). However, the logic of each PE costs too much due to the high computational complexity of message passing. To conquer the high logic cost, we propose the buffer-free PE architecture in this section.

A. Previous Work

In the message passing, both the computational complexity and logic cost are significantly affected by the model of smoothness cost *V*. Kumar and Torr [27] took advantage of a truncated model to propose a low-memory generalized BP. This reduction is effective if the convolution of (2) is fully unrolled. On the other hand, Felzenszwalb and Huttenlocher [18] reduced the message passing from $O(L^2)$ to $O(L)$ by the benefit of a linear model. Fig. 12 presents the pseudo code of their proposed message passing to calculate one new message. This code includes three loops: aggregation and

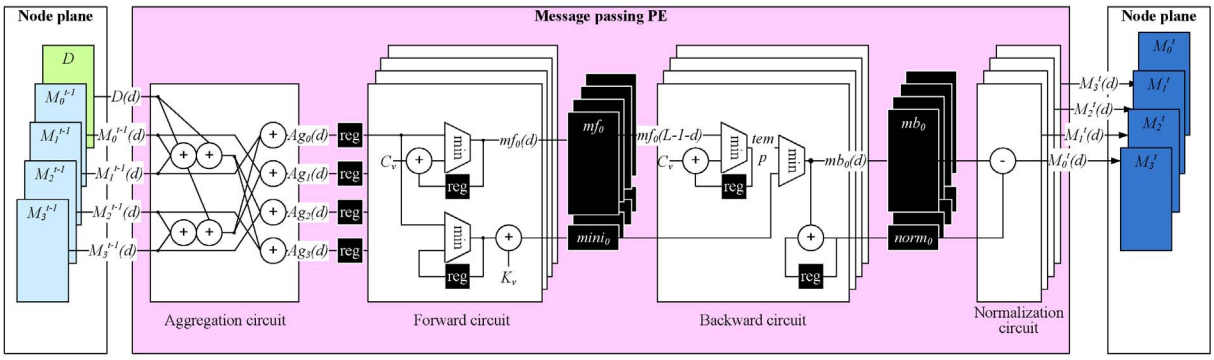


Fig. 13. Architecture of Park's PE.

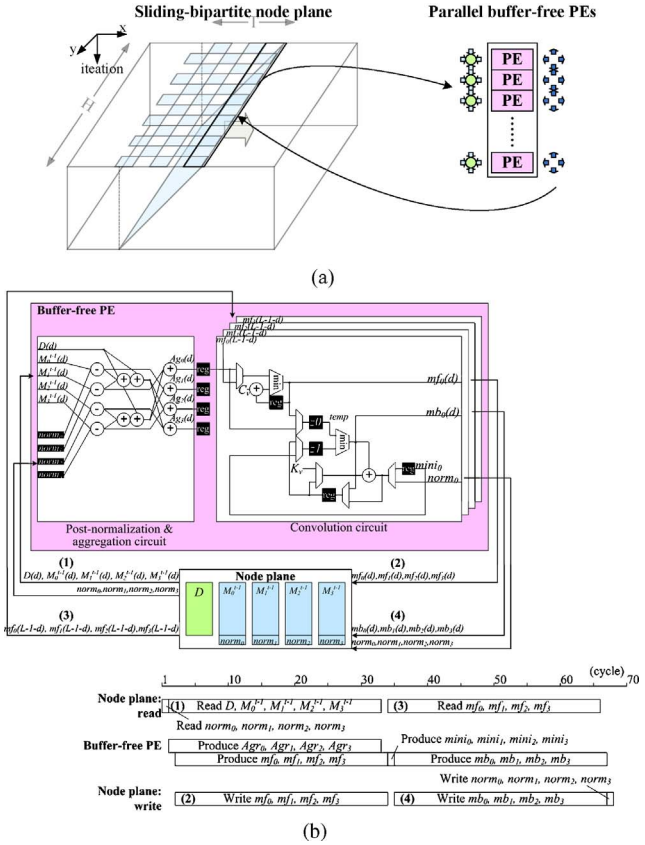


Fig. 14. Proposed architecture (a) configuration of parallel PEs on the sliding-bipartite node plane, (b) architecture of the buffer-free PE, and (c) schedule of the buffer-free PE.

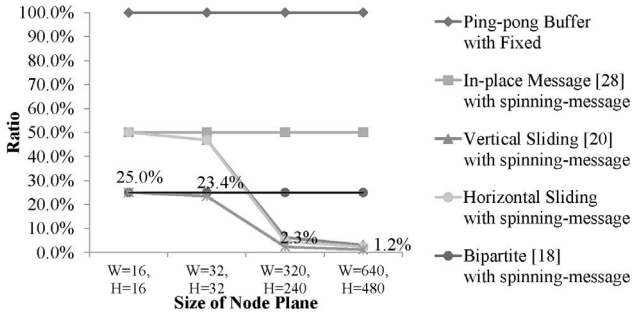


Fig. 15. Ratio of memory cost in different types of node plane with spinning-message approach.

forward process, backward process, and normalization process. The latency of each loop is L iterations.

Based on the above flow, Park *et al*. [20] directly designed a PE architecture as shown in Fig. 13. In this architecture, the node plane additionally stores the data cost. By sequential computation, four old incoming messages and data cost of a node are fetched, and four new messages of neighboring nodes are produced. This architecture uses three pipeline stages corresponding to three loops in Fig. 12. They are divided by the two large message buffers mf and *mb* with L message entries, which dominate the hardware cost of this PE.

B. Buffer-Free PE Architecture

Because the message buffers are the major logic cost of the previous PE, the strategy in our architecture is to remove all the message buffers of the previous PE. Fig. 12(a) shows the overall configuration of the parallel buffer-free PEs. The parallel PEs fetch and store data by the proposed low memory access approaches, and each buffer-free PE can calculate four messages at the same time.

Fig. 14(b) shows the detailed architecture of the buffer-free PE. Based on the pseudo code in Fig. 12, we first propose the post-normalization approach that merges the computation of the normalization on line 20 with the aggregation on line 5. The benefit of this merging is that the message buffer *mb* could be eliminated, but the *norm* storing the normalization term should be changed to node plane. It causes that the memory of each message in node plane is increased by one memory entry. Second, we propose the convolution circuit that combines the forward process on lines 6 and 7 with the backward process on lines 13 to 15. These two have identical computations, two adders and two comparators, so that these computations can share the operators with additional multiplexers for selecting data path. Thus we can remove the message buffer *mf*. Finally, we also add the pipelining registers *z0* and *z1* to cut the critical path in this architecture.

The schedule of data access and computation in the proposed PE architecture are presented in Fig. 14(c). In step (1), the normalization terms, the old messages, and the data cost are read to calculate the forward messages. In step (2), the forward messages are stored in the node plane sequentially. In

TABLE I
COMPARISON OF MEMORY COST IN MEMORY ACCESS APPROACHES FOR THE ITERATION COUNT OF 30

Type of Node Plane	Memory Access Approach	Memory Cost of Message (16 bit)	Block-Scale		Image-Scale	
			W = 16, H = 16 (kB)	W = 32, H = 32 (kB)	W = 320, H = 240 (kB)	W = 640, H = 480 (kB)
Ping-pong buffer	Fixed	*8HWL*	131	524	39 321	157 286
	Spinning-message	*4HWL*	65	262	19 660	78 643
In-place message [28]	Fixed	*4(HW + 3W)L*	77	286	19 906	79 134
	Spinning-message	*4HWL*	65	262	19 660	78 643
Vertical sliding [20]	Fixed	*8TWL*	131	491	4915	9830
	Spinning-message	*4TWL*	65	245	2457	4915
Horizontal sliding	Fixed	*8HTL*	131	491	3686	7372
	Spinning-message	*4HTL*	65	245	1843	3686
Bipartite [18]	Fixed	*4HWL*	65	262	19 660	78 643
	Spinning-message	*2HWL*	32	131	9830	39 321
Sliding-bipartite	Fixed	*4HTL*	65	245	1843	3686
	Spinning-message	*2HTL*	32	122	921	1843

step (3), the forward messages are read to calculate the backward messages. Finally, in step (4), the backward messages and new normalization terms are stored in the node plane. The memories of the node plane are implemented by two-port register files because the proposed PE read and write them at the same time. Although the proposed PE takes about double latency of the Park's PE, the logic cost has been significantly reduced because all the message buffers are removed.

The proposed buffer-free PE can compute four messages of one node at the same time. It can also compute one message of multiple nodes for different scan schemes by the following simple modification. First, the post-normalized and aggregation circuit is modified to receive three messages. Then, the convolution circuit is modified to be only one module. Finally, the accessed node plane should be properly modified according to the specific scan scheme. This modification can make the proposed PE work well for one message, but will slightly degrade the hardware efficiency due to no sharing operators in the post-normalized and aggregation circuit.

V. IMPLEMENTATION RESULT AND COMPARISON

A. Memory Cost Comparison

The memory cost is affected by the type of node plane and memory access approach. As mentioned in Section III, the type of node plane would affect the computing order of PEs, and the memory access approach would provide a data access order for node planes. Both of the type of node plan and the memory access approach do not change the computational efficiency of the message passing but the memory cost.

Table I compares the memory cost used by various types of node plane adopting the traditional fixed memory access approach and the proposed spinning-message approach. The size of node plane is substituted with block-scale and image-scale magnitudes, and each entry of messages is 16 bit. Compared to the traditional approach, most types of the node planes can save 50% memory cost in both the scales with the proposed spinning-message approach. The only exception is our previous in-place message node plane that has less saving with our approach since its original memory cost has been reduced to near 50%. In the comparison of overall hardware efficiency, the proposed spinning-message approach is better than the traditional approach. The reasons are that the proposed approach needs the same cycle counts as the traditional one while saving much memory cost. The only overhead of the proposed approach is a simple address generator, which has similar complexity as that in the traditional one.

Fig. 15 compares the memory cost among different types of node plane using the same proposed spinning-message approach for different sizes of node plane. In this figure, all the memory cost ratios are relative to the ping-pong buffer with the traditional fixed memory access approach. Compared to the sliding node planes, the bipartite node plane can save more memory cost in the block-scale. On the contrary, the sliding node planes can reduce more in the image-scale. The proposed sliding-bipartite node plane combines their benefits to reduce more memory cost in the block-scale and image-scale. Its memory cost reduction can achieve 1.2% in the image-scale of 640×480 and 23.4% in the block-scale of 32×32. Note that the sliding-based node planes would decrease its memory cost reduction when the iteration count T is larger than H or W.

B. Implementation

The proposed buffer-free PE architecture has been implemented by Verilog and synthesized by the 90 nm CMOS technology process. To compare with the Park's PE [20], we also implemented their PE design since their original implementation is on two FPGA boards. In addition, the Cheng's PE [24] is implemented in the same design condition for a fair comparison since some details are not disclosed in the paper. All the data widths are 16 bit in each implementation. Table II compares the logic cost of the proposed buffer-free PE with the other PEs. In these PEs, the Cheng's PE takes the least latency to calculate a new message because of its fully parallel architecture. The Park's PE and the proposed buffer-free PE belong to sequential architecture that causes higher latency. Although the proposed PE requires the most latency, its hardware efficiency is 3.6 times of the Park's PE and 1.4 times of the Cheng's PE. That is because we remove all the message buffers and common circuits to reduce logic cost, as well as add a pipeline stage on its critical path in the proposed buffer-free PE.

TABLE II
LOGIC COST COMPARISON OF PE ARCHITECTURES

	Cheng's PE [24]	Park's PE [20]	Proposed Buffer-Free PE	Proposed Buffer-Free PE (32 PEs)
Operating frequency (MHz)	100	222	285	285
Disparity range (L)	32	32	32	32
CMOS tech. process	UMC 90 nm	UMC 90 nm	UMC 90 nm	UMC 90 nm
Gate count (K)	69.6	50	8.3	256.6
Latency (cycle)	1 (1 msg)	32 (4 msg)	68 (4 msg)	68 (128 msg)
Throughput (node/s)	25 000K	6938K	4191K	134 117K
Hardware efficiency (throughput/gate count)	359	139	505	505

TABLE III
IMPLEMENTATION RESULTS OF VARIOUS BP-BASED ALGORITHMS FOR FRAME SIZE OF 640 × 480 AND DISPARITY RANGE OF 32

	Baseline BP [13]	Hierarchical BP [18]	Block-Based BP [25]	Tile-Based BP [22]
Iteration T	30	5, 5, 10, 5	30	Inner = 8, outer = 2
Required throughput (node/frame)	4 608 000	1 212 000	4 608 000	4 915 200
Operating frequency (MHz)	285	285	285	285
Number of PE	33	9	32	32
Gate count (K)	273.9	74.7	265.6	265.6
Size of sliding-bipartite node plane	30 × 480 (image-scale)	5 × 480 (image-scale)	30 × 32 (block-scale)	8 × 32 (block-scale)
Memory cost of messages and data costs (kB)	2793	465	186	49
FPS	30.01	31.12	29.11	27.29

Note that the hardware efficiency in our PE excludes the memory overhead by the post-normalization approach, which is highly related to the size of node plan, instead of the number of PE. Thus, our hardware efficiency will be still higher than Cheng's design when the size of node plan is smaller than 35 for one PE case. For the 32 PEs case as in Table III, the proposed approach will still have better hardware efficiency for node planes up to 35 × 32 (1120) nodes. With this size, our proposed PE is suitable for the block-scale BP algorithms, such as block-based BP and tile-based BP, whose overall cost will be more practical than that in the image-scale BP.

The proposed low memory access approach and buffer-free PE architecture could be generally applied to the various BP-based algorithms together. Table III shows the implementation results of four typical BP-based algorithms for the real-time constraint of 640 × 480 and the disparity range of 32.

In these BP-based algorithms, their algorithm flows and iteration counts affect the required throughput. The message passing is performed for the baseline BP on a whole image, and for the hierarchical BP on multiple resolution images with different iteration counts. Thus, their required throughput is proportional to the image size and corresponding iteration count. For the block-based and tile-based BP, the message passing is performed on each block (tile) in an image. In addition to the iteration count for each block, the tile-based BP has the outer iteration count for re-processing the image. Their required throughput is proportional to the total iteration count as well as the block's count and size. To satisfy the required throughput of these BP-based algorithms, we should use sufficient parallel PEs. Note that the maximal number of PE is equal to H due to the configuration of parallel PEs in the sliding-bipartite node plane. As a result, the block-based BP and tile-based BP designs just approach to real-time speed. With the buffer-free PE architecture, the logic cost of all the BP-based algorithms are less than the gate counts of 300K.

The memory cost of this table contains the messages and the data costs, which is proportional to the size of sliding-bipartite node plane according to Table I. The total memory cost of the baseline BP and hierarchical BP is larger than others because they allocate image-scale node planes. In contrast, the block-based BP and tile-based BP are more suitable to be integrated into stereoscopic video systems.

VI. CONCLUSION

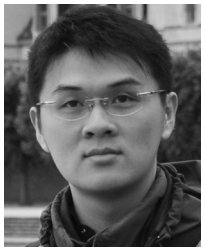
In this paper, we addressed the memory and logic cost of the message passing in BP-based algorithms. We proposed the spinning-message approach and the sliding-bipartite node plane to significantly reduce the memory cost, and the buffer-free PE architecture to reduce the high logic cost. Furthermore, the proposed low memory access approach and the buffer-free PE architecture could be applied to all typical BP-based algorithms. With the proposed approaches, the implementation of BP-based algorithms could save significantly hardware cost as well as achieve real-time speed with the same quality.

REFERENCES

[1] Philips Electronics. *Philips 3-D Solutions* [Online]. Available: http://www.business-sites.philips.com/3dsolutions/home/index.page

[2] Sharp Corporation. *Sharp Laboratories of Europe: 3-D Research* [Online]. Available: http://www.sle.sharp.co.uk/research/optical_imaging/3d_research.php

[3] iZ3D Corporation. *The World's First 3-D Monitors Designed for Gamers: iZ3D* [Online]. Available: http://www.iz3d.com
[4] M. Tanimoto, "Overview of free viewpoint television," *Signal Process.: Image Commun.*, vol. 21, no. 6, pp. 454–461, 2006.
[5] *Joint Draft 6.0 on Multiview Video Coding*, ISO/IEC JTC1/SC29 and ITU-T SG16 Q.6 JVT-Z209, Jan. 2008.
[6] *Applications and Requirements on FTV*, ISO/IEC JTC1/SC29/WG11 N9466, Shenzhen, China, Oct. 2007.
[7] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *Int. J. Comput. Vision*, vol. 47, nos. 1–3, pp. 7–42, 2002.
[8] M. Z. Brown, D. Burschka, and G. D. Hager, "Advances in computational stereo," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 8, pp. 993–1008, Aug. 2003.
[9] J. Diaz, E. Ros, R. Carrillo, and A. Prieto, "Real-time system for high-image resolution disparity estimation," *IEEE Trans. Image Process.*, vol. 16, no. 1, pp. 280–284, Jan. 2007.
[10] J. Lu, S. Rogmans, G. Lafruit, and F. Catthoor, "Real-time stereo correspondence using a truncated separable Laplacian kernel approximation on graphics hardware," in *Proc. IEEE ICME*, Jul. 2007, pp. 1946–1949.
[11] I. J. Cox, S. L. Hingorani, and S. B. Rao, "A maximum likelihood stereo algorithm," *Comput. Vision Image Understanding*, vol. 63, no. 3, pp. 542–567, May 1996.
[12] V. Kolmogorov and R. Zabih, "Computing visual correspondence with occlusions via graph cuts," in *Proc. IEEE ICCV*, Jul. 2001, pp. 508–515.
[13] J. Sun, N. N. Zheng, and H. Y. Shum, "Stereo matching using belief propagation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 7, pp. 787–800, Jul. 2003.
[14] M. Gong and Y.-H. Yang, "Real-time stereo matching using orthogonal reliability-based dynamic programming," *IEEE Trans. Image Process.*, vol. 16, no. 3, pp. 879–884, Mar. 2007.
[15] S. Park and H. Jeong, "Real-time stereo vision FPGA chip with low error rate," in *Proc. IEEE Int. Conf. Multimedia Ubiquitous Eng.*, Apr. 2007, pp. 751–756.
[16] D. Scharstein and R. Szeliski. *Middlebury Stereo Evaluation: Version 2* [Online]. Available: http://vision.middlebury.edu/stereo/eval
[17] D. Gallup, J.-M. Frahm, P. Mordohai, Q. Yang, and M. Pollefeys, "Real-time plane-sweeping stereo with multiple sweeping directions," in *Proc. IEEE Conf. CVPR*, Jun. 2007, pp. 1–8.
[18] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient belief propagation for early vision," in *Proc. IEEE Conf. CVPR*, Jun.–Jul. 2004, pp. 261–268.
[19] Q. Yang, L. Wang, R. Yang, S. Wang, M. Liao, and D. Nister, "Real-time global stereo matching using hierarchical belief propagation," in *Proc. BMCV*, 2006.
[20] S. Park, C. Chen, and H. Jeong, "VLSI architecture for MRF based stereo matching," in *Proc. Int. Symp. SAMOS*, Jul. 2007, pp. 55–64.
[21] 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. Pattern Anal. Mach. Intell.*, vol. 30, no. 6, pp. 1068–1080, Jun. 2008.
[22] C.-C. Cheng, C.-K. Liang, Y.-C. Lai, H. H. Chen, and L.-G. Chen, "Analysis of belief propagation for hardware realization," in *Proc. IEEE Workshop SiPS*, Oct. 2008, pp. 152–157.
[23] C.-C. Cheng, C.-K. Liang, Y.-C. Lai, H. H. Chen, and L.-G. Chen, "Fast belief propagation process element for high-quality stereo estimation," in *Proc. IEEE ICASSP*, Apr. 2009, pp. 745–748.
[24] C.-K. Liang, C.-C. Cheng, Y.-C. Lai, L.-G. Chen, and H. H. Chen, "Hardware-efficient belief propagation," in *Proc. IEEE Conf. CVPR*, Jun. 2009, pp. 80–87.
[25] Y.-C. Tseng, N. Chang, and T.-S. Chang, "Low memory cost block-based belief propagation for stereo correspondence," in *Proc. IEEE ICME*, Jul. 2007, pp. 1415–1418.
[26] T. Yu, R.-S. Lin, B. Super, and B. Tang, "Efficient message representations for belief propagation," in *Proc. IEEE ICCV*, Oct. 2007, pp. 1–8.
[27] M. P. Kumar and P. H. S. Torr, "Fast memory-efficient generalized belief propagation," in *Proc. ECCV*, May 2006, pp. 451–463.
[28] Y.-C. Tseng, N. Y.-C. Chang, and T.-S. Chang, "Block-based belief propagation with in-place message updating for stereo vision," in *Proc. IEEE APCCAS*, Dec. 2008, pp. 918–921.



Yu-Cheng Tseng (S'07) received the B.S. degree in electronic engineering from National Chiao-Tung University (NCTU), Hsinchu, Taiwan, in 2006. He is currently pursuing the Ph.D. degree from the Department of Electronics Engineering, NCTU.

His current research interests include 3-D video processing, and hardware architecture design and implementation.

Tian-Sheuan Chang (S'93–M'06–SM'07) received the B.S., M.S., and Ph.D. degrees in electronic engineering from National Chiao-Tung University (NCTU), Hsinchu, Taiwan, in 1993, 1995, and 1999, respectively.

He is currently an Associate Professor with the Department of Electronics Engineering, NCTU. From 2000 to 2004, he was a Deputy Manager with Global Unichip Corporation, Hsinchu. His current research interests include (silicon) intellectual property and system-on-a-chip design, very large scale integration signal processing, and computer architecture.