Hardware- and Memory-Efficient Architecture for Disparity Estimation of Large Label Counts

Sih-Sian Wu, *Member, IEEE*, Hon-Hui Chen, *Member, IEEE*, and Liang-Gee Chen, *Fellow, IEEE*

***Abstract*— Belief propagation (BP)-based stereo matching has popular owing to its regularity and ability to yield promising results. Some commonly observed hardware-implementation challenges pertaining to the use of this algorithm are large memory requirements and trade-offs between speed and chip area, along with an increasing disparity range. The paper presents a hardware- and memory-efficient architecture for building a BP-based disparity estimation system capable of overcoming issues associated with large disparity ranges. The proposed architecture is memory-efficient owing to the regularity of its underlying algorithm. In addition, the improved hardware efficiency can be attributed to processing element modifications to demonstrate shareable characteristics. Results obtained in this study reveal a 67.8% reduction in required memory corresponding to a time–area term complexity of $O(L(logL)^2)$, where L denotes the disparity range. This result is in stark contrast to the $O(L^2logL)$ and $O(L^2)$ complexities observed in extant studies. Compared to state-of-the-art implementations, the proposed architecture offers an 86.2% gate count reduction for message update units at a disparity range of 512. These results confirm the proposed architecture's suitability for use in large disparity scenarios.**

***Index Terms*— Belief propagation media (BP-M), tile-based belief propagation, memory-efficient architecture, stereo matching, disparity estimations, VLSI circuit design.**

I. Introduction

STEREO matching remains an important issue in computer vision. The demand for high-resolution depth maps continues to increase, especially from an automotive and autonomous application viewpoint. Full-HD resolution (1920 × 1080 px) can be considered is a minimum display common requirement nowadays, even as 4K2K displays gain increasing popularity. Moreover, three-dimensional display applications are becoming more prevalent by the day. Therefore, the attainment of high video and disparity resolutions with reasonable depth granularity remains an important ongoing research trend. Among available implementation platforms, application-specific integrated circuits (ASICs) and field-programmable gate arrays (FPGAs) afford high energy efficiencies, which is crucial for mobile and wearable devices. ASICs are more compact, and this makes this platform appropriate for use in wearable devices. Stereo-matching methods can be categorized as local and global, as described in [1]. In general, local methods, such as those described in [2], [3] are cost-effective. Other examples of such methods are described in [4]–[6]. There exist several algorithms, such as those referred to as non-local [7] and semi-global matching [8] methods, which reference conditional information over entire images. The loopy belief propagation (BP) [4], graph cut [5], and tree reweighted message passing [6] techniques of stereo-matching algorithms have been adequately discussed because they yield promising results. In addition, deep-learning-based methods, such as CSPN [9] and PSMNet [10], have recently attracted significant research attention. They benefit from being able to adapt to learning and inferencing data domains. However, Suleiman *et al.* [11] demonstrated that handcrafted features are up to ten times as energy-efficient as learned features. The existence of a large disparity is a major challenge facing hardware implementations in terms of both memory utilization and hardware efficiency. Memory size is directly proportional to the disparity range L, a large value of which implies the accurate representation of 3D information, which is a crucial requirement in high-resolution applications.

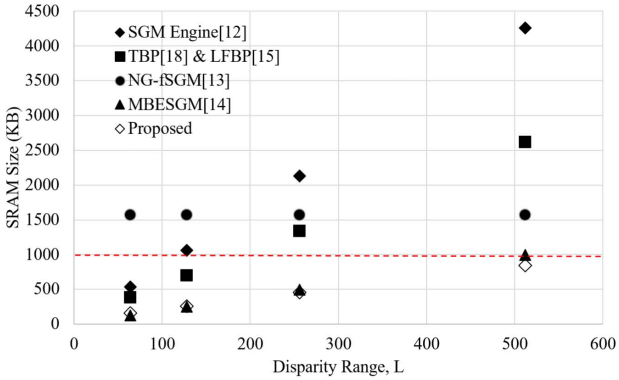


Fig. 1. SRAM sizes of different architecture and depth labels. The SRAM sizes of the SGM engine [12], NG-fSGM [13], and MBESGM [14] are simulated based on their reports.

Recently, light-field belief propagation (LFBP) [15], the semi-global matching (SGM) engine [12], and NG-fSGM [13] have been proposed as dedicated hardware for stereo matching with integer disparity ranges of 64, 128, and 128, respectively. In these designs, SRAM occupies 50% of the available area; hence, addressing large disparity scenarios is considerably challenging. Memory- and bandwidth-efficient

Manuscript received June 2, 2020; revised September 11, 2020; accepted November 1, 2020. Date of publication November 26, 2020; date of current version September 3, 2021. This article was recommended by Associate Editor Y. Wang. *(Corresponding author: Sih-Sian Wu.)*

The authors are with the Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 106, Taiwan (e-mail: benwu@video.ee.ntu.edu.tw).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCSVT.2020.3040774.

Digital Object Identifier 10.1109/TCSVT.2020.3040774

SGM (MBESGM) [14] features a memory-efficient architecture with the incompleteness and truncated techniques. Large numbers of disparity candidates are all used in the previous architecture to compute the optimization process, resulting in significant computational complexity. Since computation complexity increases with disparity, hardware efficiency becomes critically important in large disparity stereo-matching hardware designs. The BP algorithm is the most regular in computation; therefore, making it hardware-friendly, and suitable for very large-scale integration (VLSI) implementation. However, there exist two problems associated with the VLSI implementation of BP algorithms. The first is a serious memory problem, which causes a major obstruction to VLSI implementation [16]. Message-compression techniques have been proposed by Yu *et al.* [17] to offset this problem. Block- [16] and tile-based [18] BPs have also been proposed to significantly reduce memory demands. Fig. 1 compares the memory requirements of different algorithms. Except for MBESGM [14], all these existing architecture require more than 1, 500 kB of SRAM to support scenarios with a depth label of 512. A tile-based BP disparity-estimation system has been proposed in [15] for full-HD (1920×1080 px) resolution displays with a disparity range of 64. It exemplifies the severity of the memory problem, as over 50% of the chip area is occupied by SRAM. This problem becomes more challenging with increasing disparity range. Along with large memory consumption, an increasing disparity range necessitates a tradeoff between speed and hardware complexity, as described in Table I.

Major contributions of the proposed architecture include

1) Efficient hardware implementation of large label-count belief propagation;
2) Memory-efficient message passing; and
3) Fast and hardware-efficient message updating.

The remainder of this manuscript is organized as follows. Section II addresses the problem at hand and briefly reviews several extant works. The proposed architecture is discussed in Section III. Experimental results, discussion, and conclusions drawn from this research are presented in Sections IV–VI, respectively.

II. Previous Work

A. Global Methods Overview

In general, global methods outperform local methods. Global algorithms optimize a Markov random field (MRF) using techniques such as graph cuts, tree-reweight message passing(TRW-S) [20], (loopy) belief propagation (BP), and SGM [12]. Among these, BP and SGM are considered more hardware-friendly owing to their demonstrated regularities. One of the more critical problems associated with global methods is computation complexity. GPU [21], FPGA [22], [23], and ASIC implementations [15], [18], [24] have been proposed to tackle these problems. The SGM engine [12] features partitioned progress that only stores in-block aggregated cost in the on-chip SRAM. Neighbor-guided SGM (NG-fSGM) [13] selects particular candidates randomly and candidates guided from neighboring pixels, which makes the computation and SRAM size independent of the label counts. However, it requires more than 1, 500 kB SRAM for processing under a disparity range of 512. MBESGM [14] employs incompleteness and inaccuracy techniques to further reduce on-chip memory requirements. As all disparity candidates are processed, computations under a large disparity range become impractical. Thus, an optimal architecture for large label counts should account for both on-chip memory size and computational complexity. Because mobile devices require high energy efficiencies, ASIC implementations are considered more appropriate in mobile applications. This study considers BP as the target algorithm owing to its regular data flow and simple processes. Both features are important from a hardware-implementation viewpoint.

B. Belief Propagation Implementation Overview

BP is a global energy minimization algorithm for labeling problems, and it has been employed in such applications as disparity search [25] [26] [18], optical flow [26], denoising [25], image inpainting [18], image registration [27], and structure from motion [28]. The BP energy function comprises unitary and pairwise terms described as follows.

$$\{l\} = \underset{l_p \in L}{\operatorname{argmin}} \left\{ \sum_{p \in P} E_d\left(l_p\right) + \sum_{(p,q) \in N_p} E_s\left(l_p, l_q\right) \right\}, \quad (1)$$

Here, l, l_p and l_q denote the generated label with lowest energy, label value of the pixel p, and label value of neighboring pixel q, respectively. Some studies identify the unitary and pairwise terms as data and smoothing terms, respectively. The proposed architecture is inspired by a tile-based BP approach [18], which is one of the more suitable architectures for BP hardware implementation with reasonable on-chip memory requirements.

Fig. 2 depicts a block diagram of the proposed architecture comprising a cost-computation processing element (PE) pool, message update PE pool, message buffer, and cost buffers. The message and cost buffers are stored in on-chip SRAM. The memory- and hardware-efficiency issues are addressed in the message-passing data flow and message-updating blocks, respectively.

The BP energy function can be expressed as

$$E(f) = \sum_{(p,q) \in N} V(f_p, f_q) + \sum_{p \in P} D_p(f_p) \quad (2)$$

1) Message Passing Data Flow: A BP-based disparity estimation algorithm, BP media (BP-M) [4], is used widely as a depth estimation method, providing a global energy minimization solution to the stereo matching algorithm. The concept of BP-M is shown in Fig. 4; each node receives messages from each of four neighbors and maintains its own data cost. Eq. 3 illustrates the message passing from the node (i, j) to the node $(i + 1, j)$. After the message updating block processes the message $M^*_{Left}(i; j)$, it is translated into $M_{Left}(i; j)$ as shown in Eq. 4, where $M_{Left}(i; j)$ represents the message from left for node (i, j) and $C(i; j)$ denotes the data cost of a specific pixel. There are four direction message

TABLE I

SUMMARY OF VARIOUS ARCHITECTURES. L IS THE DISPARITY RANGE AND T IS THE TRUNCATED NUMBER FOR THE SMOOTHING TERM

Categories	Path-based		Tree-based		
Architecture	EBP[19]	LFBP[15]	BP-M[4]	TBP[18]	Proposed
Feature	Higher hardware efficiency		Low critical path		Low critical path and Higher hardware-efficiency
Critical Path	$O(L)$	$O(L)$	$\boldsymbol{O(log_2 L)}$	$\boldsymbol{O(log_2 L)}$	$\boldsymbol{O(log_2 L)}$
Area	$\boldsymbol{(L)}$	$\boldsymbol{O(L)}$	$O(L^2)$	$O(L^2)$	$O(Llog_2 L)$
Hardware-Efficiency	$O(L^2)$	$O(L^2)$	$O(L^2 log_2 L)$	$O(L^2 log_2 L)$	$\boldsymbol{O(L(log_2 L)^2)}$

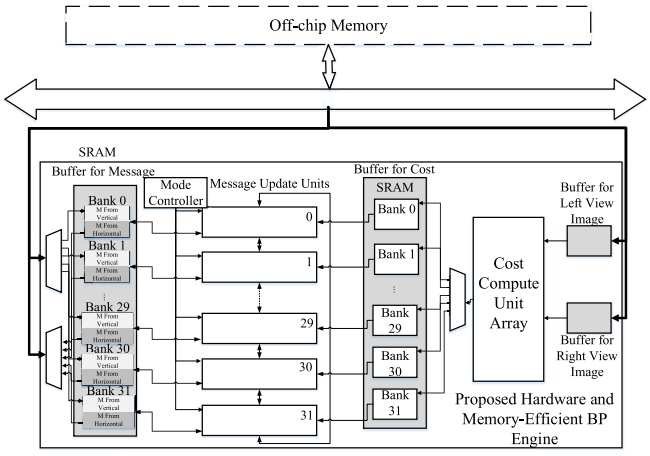


Fig. 2. **Block diagram of the proposed belief propagation engine.** This engine consists of three components: the cost computation PE pool, the message update PE pool, and the message buffer and cost buffer (The message buffer and cost buffer are stored in SRAM).

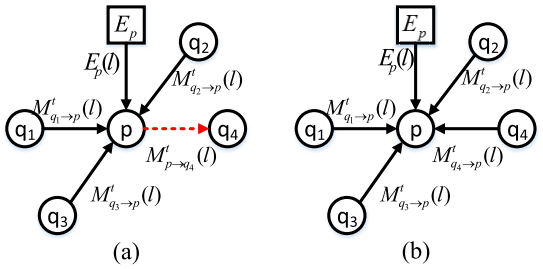


Fig. 3. (a) The passing mode and (b) the deterministic mode for a BP algorithm.

sets and one cost set for each pixel and each set consists of L elements. BP-M requires $O(4WHL)$ SRAM, where W, H, and L are the width, height, and disparity range of the image, respectively. Tile-based approaches such as [18] and [15], and block-based approaches [16] keep partial data in SRAM only which reduces the size to $O(4T^2L)$, where T is the diameter of the tile or block size. Although constant-space BP [29] has been proposed to solve the memory issue, its memory size is irrelevant, relative to the disparity range, as it requires a further process to reconstruct the original energy function. The disparity range related reconstruction process makes this implementation unsuitable for large disparity scenarios. The above-mentioned idea is inspired by the hierarchy belief propagation [30]. The message updating block will be introduced in the next subsection. Although each row can

Algorithm 1 Algorithm for Message Passing

Input: $C_1, C_2, \ldots, C_n$, $M_1^{Left}, M_2^{Left}, \ldots, M_n^{Left}$, $M_1^{Right}, M_2^{Right}, \ldots, M_n^{Right}$, $M_1^{Up}, M_2^{Up}, \ldots, M_n^{Up}$, $M_1^{Down}, M_2^{Down}, \ldots, M_n^{Down}$ and Passing direction $D \in \{Left, Right, Up, Down\}$

Output: $M_1, M_2, \ldots, M_n$

for $j = 1; j \leq n$ **do**

  $M_j = C_j + \sum_{d \notin D} M_j{}^d$;

end

return $M_1, M_2, \ldots, M_n$;

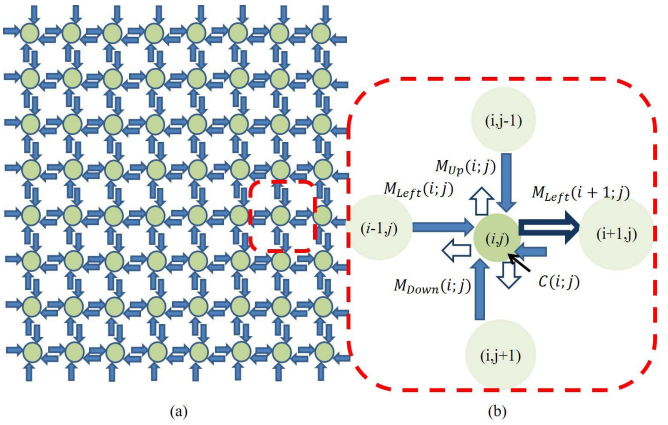


Fig. 4. (a) Framework of BP-M. (b) Detailed message flow of a single node.

process simultaneously during horizontal processing and vice versa, there exists a high data dependency between adjacent messages.

$$M_{ToPE}^{Left}(i;j) = M^{Left}(i;j) + M^{Up}(i;j) + M^{Down}(i;j) + C(i;j) \quad (3)$$

$$M_{ToPE}^{Left}(i;j)' = MessageUpdate(M_{ToPE}^{Left}(i;j)) \quad (4)$$

The conventional message passing data flow is shown in Alg. 1. In the BP-based disparity estimation, each pixel receives four messages from up, down, left, and right neighboring nodes. The relationship between each pixel and its messages is shown in Fig. 4. In the original BP-M [4] system, the four direction messages for every pixel label in the entire set of images are stored in individual memory spaces. The memory requirements are immense and proportional to the image size and the disparity range, L.

Block-based BP [16] has been proposed to solve the large memory problem by storing messages of a processing block only. The decrease in performance is severe because of the lack of information outside the processing block itself. This technique disconnects the SRAM demands from the size of the input image. Liang *et al.* [18] proposed tile-based BP (TBP), which is derived from the concept of the block-based BP [16] with boundary message sharing. Moreover, it reduces the decrease in performance of the block-based BP [16]. A few researchers have leveraged the hardware-friendliness of tile-based BP to implement their proposed systems on GPU or FPGA platforms [31]. VLSI implementations have also been proposed in [32] and [15]. Similar to the block-based BP, the tile-based BP approach stores the cost value and messages of the processing tile in the on-chip SRAM while maintaining the boundary information in the off-chip DRAM. Although the SRAM size is reduced, it still dominates the chip size. In a state-of-the-art approach, [15], over 50% of the chip area is occupied by the SRAM, a problem which becomes more severe when the disparity range increases. In the proposed system, we adopt a memory-efficient technique that reduces the memory size by over 69.5% for $L = 512$.

2) Message Update Unit: We follow the notation used in [18] in the remainder of this paper. To find the minimum energy for each disparity level, the cost function used is

$$M_{pq}^{t}(l) = \min_{l' \in L} \left\{ E_s(l, l') + E_d(l') + \sum_{p' \in N_p \setminus q} M_{p'p}^{t-1}(l') \right\}. \quad (5)$$

where $M_{pq}^{t}(l)$ is the message passed from pixel p to pixel q in $t-th$ iteration at level l. The terms that are independent of l provide the initial guess for the iterative process. They are combined as $H\left(l'\right)$ which is defined as

$$H\left(l'\right) = E_d(l') + \sum_{p' \in N_p \setminus q} M_{p'p}^{t-1}(l'). \quad (6)$$

The term E_s is the smoothness penalty. According to its regularity, the truncated linear model [19] is friendly to hardware implementation [18] [19] [15] and is defined as

$$E_s\left(l, l'\right) = \min_{l' \in L}(\lambda \left| l - l' \right|, \lambda T), \quad (7)$$

where λ is the weight of the smoothness and T is the parameter that constrains the quantity from increasing. For simplicity, Eq. 5 can be transformed into

$$M_{pq}^{t}(l) = \min_{l' \in L} \left\{ E_s(l, l') + H\left(l'\right) \right\}. \quad (8)$$

The conventional message update process is shown in Alg. 2. The computational complexity of traditional hardware implementations (see [4]) is $O(L^2)$ where L is the disparity range. This approach is not practical to implement in hardware when the label counts L is large, as shown in Fig. 6(a). An efficient BP implementation, such as [19], has been proposed to improve the hardware efficiency of our architecture, which we will refer to as efficient BP (EBP). This two-pass method is adopted to find the minimum envelope of the disparity set. Even though this approach is more efficient, its critical path is directly proportional to the disparity range as shown in Fig. 6(b). This makes EBP unsuitable for large disparities. To shorten the critical path, a modification in [15] removes the backward pass of the two-pass method, making its critical path half the size of EBP. This architecture is implemented in a seven-stage pipeline with a cost of performance loss as shown in Fig. 6(d). However, the approach in [15] becomes impractical when the label count is larger than 64. Existing methods can be classified into two categories: tree-based and path-based methods. Tree-based methods [4], [18] have shorter delays but worse hardware-efficiency. Compared to tree-based methods, path-based methods [15], [19] attain lower hardware complexities, but come with longer delays, which are proportional to the disparity range. Chang *et al.* [33] derived a fast belief propagation (FBP) architecture using the conventional implementation and the following formula:

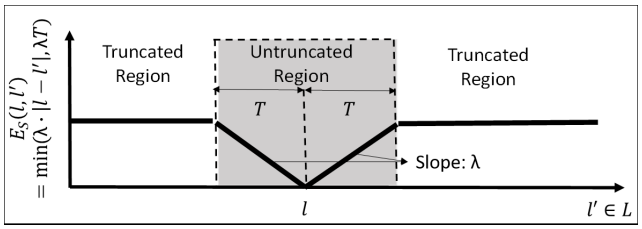


Fig. 5. Smoothing term with the truncated linear model.

$$M_{pq}(l) = min\left(M_{pq}^{Local}(l), M_{pq}^{Global}\right). \quad (9)$$

Here, $M_{pq}^{Local}(l)$ is the minimum value within the non-truncation region, which is computed in local trees.

$$M_{pq}^{Local}(l) = \min_{l-T \le l' \le l+T} \left\{ E_s(l, l') + H\left(l'\right) \right\}. \quad (10)$$

M_{pq}^{Global} is the global minimum constraint, which is independent of the disparity level computed in the global tree.

$$M_{pq}^{Global} = \min_{l' \in L} \left\{ T\lambda + H\left(l'\right) \right\}. \quad (11)$$

In [18], the designed structure was efficient and exhibited highly parallel properties as shown in Fig. 6(c). Based on these two properties, we derived our architecture starting from [18].

III. Proposed System

This chapter introduces the proposed VLSI design for stereo matching. This system comprises an efficient message updating PE pool, a matching cost computation PE pool, and buffers for the messages and pixels of the input images. Because BP hardware implementations generally suffer from large on-chip memory and an inefficient computation when the disparity range is large, our goal was to develop a memory-efficient message passing technique and a hardware-efficient message updating method.

A. Memory-Efficient Data Flow

A memory-efficient data flow for message passing is involved in our proposed system to solve the large memory problem. We begin by describing the observations of our analysis. Thereafter, we present our memory-efficient data flow and derive the corresponding memory banking technique.

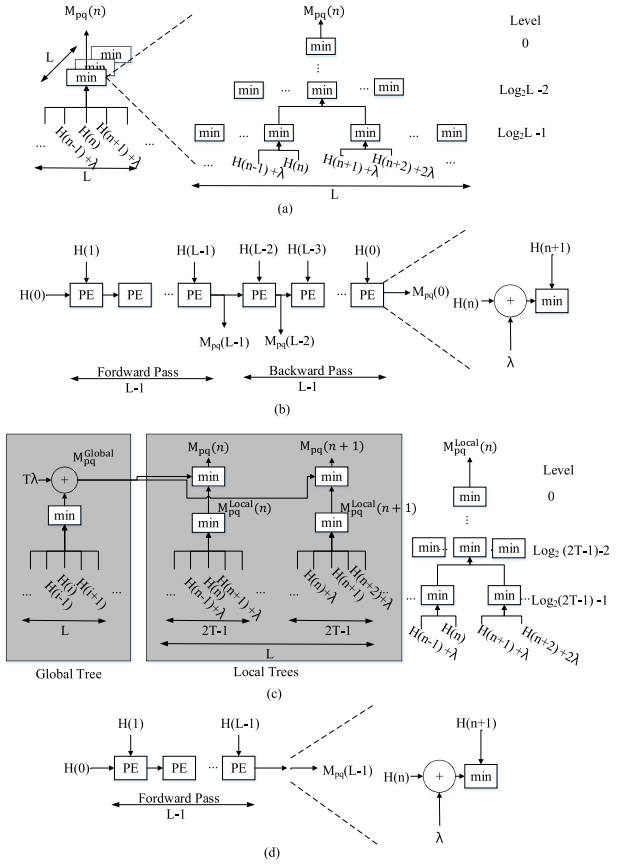


Fig. 6. System structure for (a) original hardware implementation BP-M [4], (b) efficient hardware implementation EBP [19], (c) proposed architecture in TBP [18], and (d) architecture in LFBP [15].

Algorithm 2 Algorithm for Message Update

Input: $M_1, M_2, \ldots, M_n$ and smooth equation $V(d_x, d_y)$
Output: $M_1^{New}, M_2^{New}, \ldots, M_n^{New}$

$con(r_i) = \Phi$;
for $j = 1; j \leq n$ **do**
  float $minSmooth = 999.0$;
  float $minLabel = 0$;
  float $TempSmooth = 0$;
  for $j = 1; j \leq n$ **do**
    $TempSmooth = M_i + V(d_i, d_j)$;
    if $(minSmooth) \geq (TempSmooth)$ **then**
      replace $minSmooth$ with $TempSmooth$;
      replace $minLabel$ with i;
    end
    $M_j^{New} = minSmooth$;
  end
end
return $M_1^{New}, M_2^{New}, \ldots, M_n^{New}$;

1) Analysis of a Message Update Module's Data Flow: The original data flow for the message update modules is shown in Fig. 7(a). We adopt the definitions employed in TBP [18]; thus, the BP processing order is rightward, leftward, downward, and upward. Cost values do not change during processing as shown in Eq. 3. To reduce the SRAM size, cost values are computed on-the-fly just as in [15] and [18]. For this reason, we focus on reducing message storage in this paper.

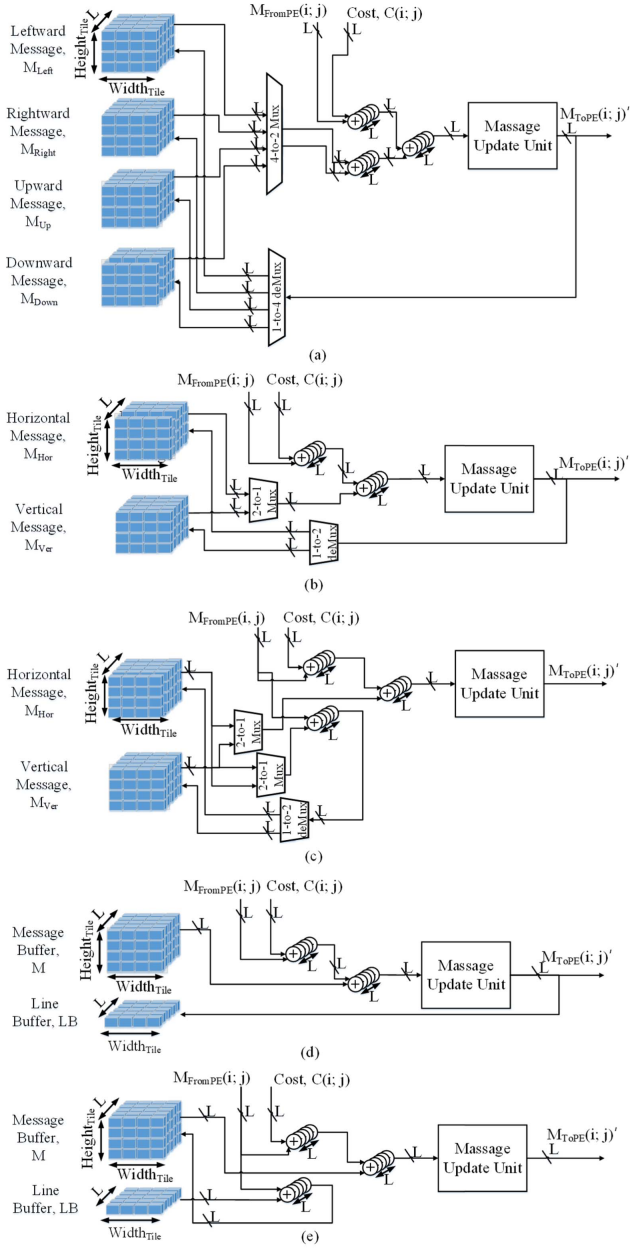


Fig. 7. **Data flow comparison of various implementations.** (a) the one of conventional BP engine, (b) the forward passing of MEMP [34], (c) the backward passing of MEMP [34], (d) the proposed forward passing, and (e) the proposed backward passing.

We summarize our observations in two ways. First, a message still occupies the memory because the updated message of any particular direction has not been generated yet, although that message is never used again. Both, BP-M and tile-based architecture replace old messages with corresponding new messages. Second, messages from the vertical direction are terminated at the same time while finishing horizontal processing and verse visa. In addition to combining messages within the same group, vertical and horizontal groups are

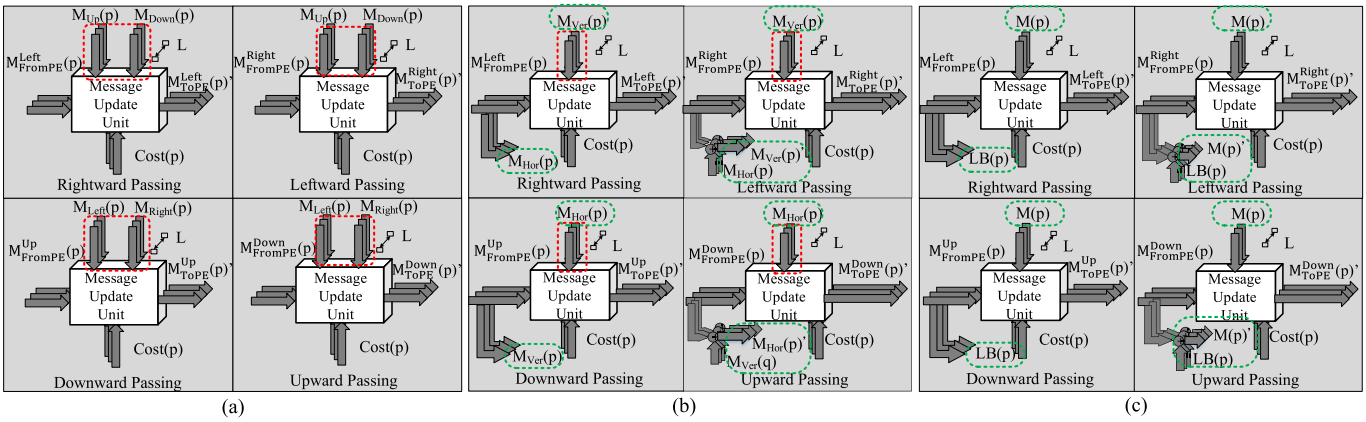


Fig. 8. **Input and output for a message update module.** (a) Conventional one in TBP [18], (b) grouping messages into horizontal and vertical, and (c) the proposed data flow.

able to share the same memory location without conflict. We followed the aforementioned observation to re-design the data flow.

2) Proposed Architecture: The original data flow of TBP [18] is shown in Fig.7(a). Based on the analysis of MEMP [34], messages in the same group are terminated at the same time. First, we separate messages into vertical and horizontal groups as Eq. 12

$$\begin{cases} M_{Ver.} = M_{Up} + M_{Down} \\ M_{Hor.} = M_{Left} + M_{Right} \end{cases} \tag{12}$$

Through this, the memory requirement can be reduced by half. However, there are two problems encountered when implementing the above architecture: how to combine the messages in the same group, and when to process them. Messages in the same group cannot be added up directly because they are generated at different times. Through timing analysis, we also notice that messages within the same group are used at the same time in an orthogonal direction process but not a parallel one. In other words, those messages can be added up only after parallel processing. Before backward processing, we have to store another message into the buffer. In this paper, we follow the definitions employed in TBP [18]. In horizontal processing, we call a processing to the **right** a *forward* passing and to the **left** a *backward* passing. Similarly, in vertical processing, we call processing from the top (respectively, down) to the down (top) as forward (backward) passing. Combining two messages into one group reduces the requirement by half, thus we can gain the benefit of summing up two messages and storing the result in memory. The above-mentioned architecture is shown in Figs. 7 (b) and (c).

Furthermore, we also observed that vertical and horizontal messages overlap in the timing slot, causing the proposed system to allocate two memory spaces to each message. We further consider memory reduction by eliminating the timing overlap. If we can postpone saving time one cycle after a backward passing, another message space is then unnecessary. By applying the above scheme, two messages can be stored in the same memory without any conflict. During forward processing, the message from the last pixel, $M_{FromPE}(i;j)$, is added to its data cost, $C(i;j)$, and messages from orthogonal directions, $M(i;j)$. Consider rightward passing, for instance, the updated message set, $M_{ToPE}(i;j)'$, passed to the next pixel, $M_{FromPE}(i;j)$, is stored in the line buffer, $LB(j)$, in the same cycle. In the backward process, the substitution of the combined message, $M(i;j)$, is processed except the summation and message update in the forward one. $M_{FromPE}(i;j)$, is added to the value in the line buffer, $LB(j)$, and then replaces the out-of-date combined message, $M(i;j)$, with the updated combined one, $M(i;j)'$. Finally, the computation for the proposed BP reduces to Eq. 13. Compared to the conventional message update equation, Eq. 5, memory buffers are reduced from four tile buffers to one tile buffer and four line buffers. An analysis of the lifetime is shown in [34], and the detailed architecture is shown in Figs. 7(d) and (e).

$$\begin{cases} \text{Forward:} \\ LB(j) & = M_{FromPE}(i;j) \\ M_{ToPE}(i;j) & = M(i;j) + M_{FromPE}(i;j) + C(i;j) \\ M_{ToPE}(i;j)' & = MessageUpdate(M_{ToPE}(i;j)) \\ \text{Backward:} \\ M_{ToPE}(i;j) & = M(i;j) + M_{FromPE}(i;j) + C(i;j) \\ M_{ToPE}(i;j)' & = MessageUpdate(M_{ToPE}(i;j)) \\ M(i;j)' & = M_{FromPE}(i;j) + LB(j) \end{cases} \tag{13}$$

The parallelism of the proposed architectures is another issue we are concerned with. Based on the analysis, this system requires more than four PEs working simultaneously to reach the desired specification, i.e., full-HD at 30 fps. The parallelism of the proposed architecture is similar to those of TBP [18] and LFBP [15]. That means the parallelism is equal to the width or the height of the tile as in the LFBP [15] mentioned above.

An SRAM banking is proposed in [15] to increase the SRAM accessing bandwidth, as shown in Fig. 9(a). To adopt the designed memory-efficient data flow, we propose a memory banking technique as shown in Fig. 9(b). The timing diagram is shown in Fig. 9(c) and (d). In forward passes, the message block buffer, M, is read, and the updated message

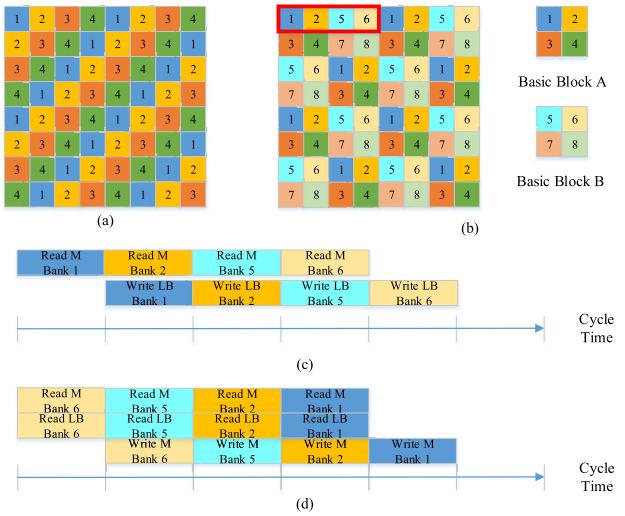


Fig. 9. **Memory banking technique of the proposed architecture.** (a) is presented in [15], (b) is proposed for this architecture, (c) is a forwarding passing timing diagram of blocked pixels in (c), and (d) is a forwarding passing timing diagram of blocked pixels in (c).

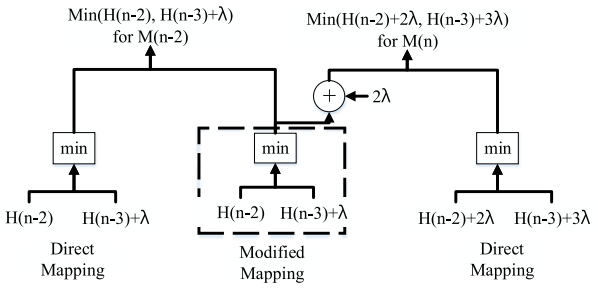


Fig. 10. Original mapping and modified mapping for different minimum operations.

is stored in the corresponding line buffer, LB, in the next cycle. In backward passes, the message block buffer, M, and the line buffer, LB, are read; the merged and updated messages are then stored in the message block buffer, M, in the next block. This module solves the large on-chip memory issue but does not address the computational inefficiency. In the next section, a faster, more hardware-efficient message updating module is proposed.

B. Fast and Hardware-Efficient Message Update

The message update process is the most computationally intensive part of a BP engine. As mentioned in the previous section, this module updates a set of messages from three incoming message sets and a matching cost set where the size of each set is the disparity range, L. BP implementations [18] [15] [19] suffer from hardware inefficiency for large disparity scenarios. In this section, we utilize regularity to design a message update PE that is fast and hardware efficient.

1) Observations: Each iteration has four direction passes for each pixel. The process updates a set of messages whose sizes are equal to the disparity range, L. Each updated message is the minimum value for L candidates, as given Eq. 9. Unlike a stereo matching criterion, the disparity with sub-pixel accuracy [26] [18] and another applications [25] is inappropriate to limit untruncated regions to such small ranges [18] while applying BP algorithms. To preserve the performance, we follow the suggestion from [29] setting $T = \frac{L}{8}$. That is, T and L are in direct proportion. The smoothing effect is unapparent, and the advantage of BP optimization is lost if T is unreasonably limited. Depth maps generated by different architectures are provided in Section IV. When the setting for T is not much smaller than L, the complexity of [18] is $O(L^2)$ instead of $O(L)$. Among smoothing functions discussed in EBP [19], the truncated linear model makes the implementation more regular. We utilize the regularity to design the proposed architecture reusing compare results as LFBP. Unlike existing hardware implementations, minimum values are computed by individual binary tree structures as shown in Fig. 6(a) and (c). In the conventional architecture, each local tree is independent of the others. Based on the truncated linear model, the difference between adjacent messages, except those belonging to the truncated region, is equivalent, as follow:

$$\begin{aligned} &min(H(n-2)+2\lambda, H(n-3)+3\lambda) \\ &\quad = min(H(n-2), H(n-3)+\lambda)+2\lambda. \end{aligned} \tag{14}$$

We derive our architecture by exploiting the regularity to reduce the hardware complexity. Modified mapping operators are shared among different local trees. The majority of minimum operators inside local trees can be removed as shown in Fig. 10. The critical path is not affected because the delay is dominated by the global tree, as shown in Fig. 6(c). The regularity property is only sustained within the same side of the untruncated region.

$$min(H(n-1)+\lambda, H(n)) \neq min(H(n-1), H(n)+\lambda) \tag{15}$$

To further increase the hardware efficiency, both sides need to be designed separately. The proposed architecture is presented in the next section.

2) Proposed Architecture: We derive the proposed hardware-efficient architecture [35] in three steps. First, reusable elements are allocated at the leaf level of all local binary trees reducing the complexity by half. Second, reusable elements are allocated at all levels of local binary trees, as shown in Fig. 11(c). The complexity of a local binary tree is reduced from $O(T)$ to $O(2log_2T)$. Finally, the architecture is designed as interleaved to further decrease the complexity by half. The complexity of a local binary tree reduces from $O(2log_2T)$ to $O(log_2T)$. The derived procedure is shown in Fig. 11.

First, reusable components are adopted at the lowest level of the local binary tree because, from the above-mentioned observation, every comparator at the lowest level can be eliminated, except for two comparators near the middle. Thereafter, the computational complexity of the local binary tree is reduced from $O(2T)$ to $O(T)$, as shown in Fig. 11(b). Then, reusable components are adopted at every level of the local binary tree. We extend the above concept to more than one

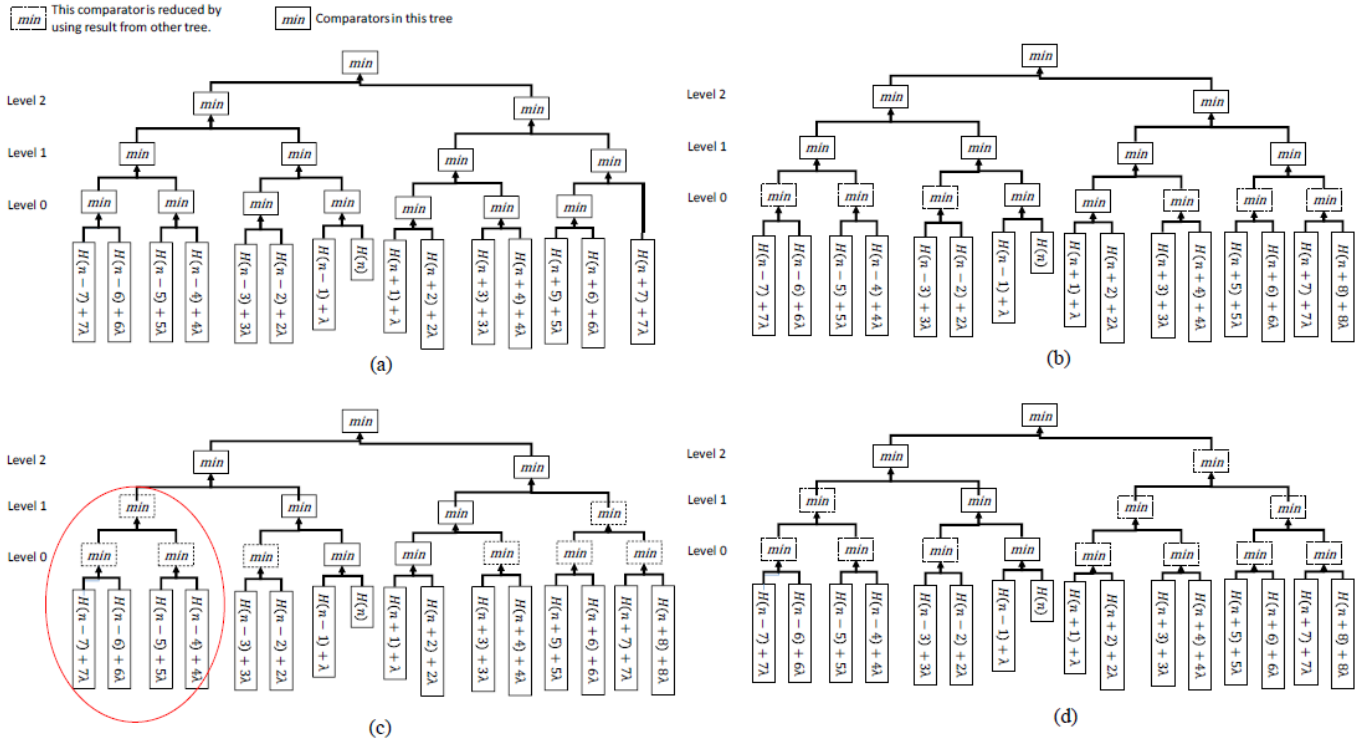


Fig. 11. Local trees structures (a) an original local binary tree in [15], (b) a binary tree after allocating reusable elements in the lowest level, (c) a binary tree after allocating reusable in all levels, and (d) the proposed architecture with interleaving.

component as given in Eq. 16.

$$\begin{aligned} &min(H(n-4)+4\lambda, H(n-5)+5\lambda, H(n-6)+6\lambda, \\ &\qquad H(n-7)+7\lambda) \\ &= min(H(n-4), H(n-5)+\lambda, H(n-6)+2\lambda, \\ &\qquad H(n-7)+3\lambda)+4\lambda. \end{aligned} \tag{16}$$

Based on the property of the truncated linear model, each minimum search operation for neighboring components can be transformed to facilitate hardware implementation and increase efficiency. In our example in Eq. 16, the original form can transform to another one with a difference 4λ, which is generated at Level 1 of the local tree $n-4$, depicted as the red circled sub-tree in Fig. 11(c). We further extend the concept to higher levels. Beyond this step, the complexity of local trees is reduced from $O(T)$ to $O(2log_2T)$, as shown in Fig. 11(c).

Finally, an interleaving technique is adopted in the proposed architecture. The interleaving procedure is shown in Fig.12. In the example, the left sub-trees of the local tree n and $n+1$ are equivalent but with λ difference. The same is found in the right sub-trees. We adopt this attribute to further reduce the complexity of the local tree from $O(2log_2T)$ to $O(log_2T)$.

Leaves in each local tree are of the size of the untruncated region, $2T-1$, which means there are $\lfloor \frac{2T-1}{2} \rfloor$ operators in the lowest level within a traditional local tree. Based on the properties explained above, we can see that the number of min operations in the bottom level is $2 \times \frac{L}{2}$ for two sides of an untruncated region. Each local tree contributes to one operator and collects other information from other trees.

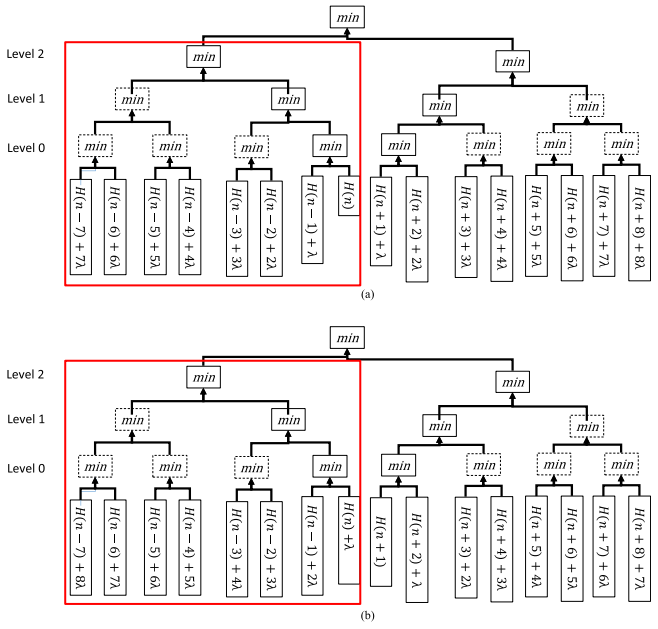


Fig. 12. The interleaving procedure for the proposed architecture. The left sub-tree of adjacent trees can be reused. (a) The local binary tree n and (b) the local binary tree $n+1$. It should be noted that both left sub-trees of the local tree n and $n+1$ are equivalent but with λ difference.

At the upper level, regularity is preserved. A conventional architecture requires $\lfloor 2 \times \frac{2T+1}{2^2} \rfloor$ operations in each local tree while the proposed one requires $2 \times \frac{L}{2}$ operations for "all" local trees. Interestingly, the total number of operators for

local trees is equal at each level. Each local tree contains only one min operator at each level. The height of a local tree is $\lceil log_2(2T+1) \rceil$.

That is, the number of total operators inside a local tree is equal to the height of the local tree, $\lceil log_2(2T-1) \rceil$. We mentioned that for $T \propto L$, the complexity of a BP processing element is reduced from $O(L^2)$ to $O(Llog_2(L))$. The problem is how to design the local tree to reach the efficiency we desire.

By contrast, the proposed local tree architecture utilizes the aforementioned property, as shown in Fig. 11(d), instead of direct mapping [18], as in Fig. 11(a). In each local tree, only one minimum operation is required at a single level. In Fig. 11, we use $T = 7$ as an example to demonstrate the difference between the original architecture and the proposed one. In the figure, we color operators in other local trees. In the given instance, the number of min operators in the original architecture and that of the proposed one are fifteen and four, respectively. To further increase hardware efficiency, the proposed local trees are divided into even and odd groups. Because of the left-right-side inequality, each group contributes to the opposing side. As a result, the number of required min operators for each local tree are reduced from $2T$ to $\lceil log_2(2T+1) \rceil$. When the value of T is much smaller than L, such as for $T = 1$ or $T = 2$, the complexity is reduced from L^2 to $Llog_2(L)$. Despite the fact that the proposed local tree has a longer critical path than the one in [18], the total delay is dominated by the global tree. The proposed architecture does not introduce a delay overhead to the whole BP processing unit.

One may notice that the numbers of leaves between the conventional and the proposed architecture in Fig. 11 are not equal. We added additional nodes in the leaves of the proposed local trees, which enhance the efficiency of the architecture. The proposed local tree contributes a min operator per level and collects information provided by other trees. More precisely, each local tree collects information from the lower level of other trees. The information from other local trees represents the minimum value among hypotheses, where the number of hypotheses is a power of 2. To fully utilize this information, we add an additional term that is not inside the untruncated region. This term makes the local tree a complete binary tree facilitating the hardware regularity. With this modification, the Eq. 9 can be rewritten as

$$M_{p\to q4}(l) = min\left(M^{Local}_{p\to q4}(l), M^{Global}_{p\to q4}, H(n \pm (T+1)) + (T+1)\lambda\right). \tag{17}$$

The additional term, $H(n \pm (T+1)) + (T+1)\lambda$, is smaller than $M^{Global}_{p\to q4}$ defined in Eq. 11. This term is added for further hardware-efficiency and without affecting the correctness.

C. Implementation Details

We added weights based on color differences in messages which is inspired by HBP [30]. Our color weight is defined as

$$\delta = 1 - \frac{|I(s) - I(t)|}{256}, \tag{18}$$

TABLE II

ERROR RATES OF DIFFERENT ARCHITECTURE WITH DIFFERENT DATASETS. THE ERROR RATE OF KITTI 2015 [37] IS 3-PX ERROR AND THE ONE OF MIDDLEBURY V3 [36] IS WITH THRESHOLD 2.0. BOTH VALUES ARE LOWER THE BETTER

Dataset	EBP [19]	FBP [33]	Ours
KITTI 2015 [37]	7.89	17.23	7.39
Middlebury V3 [36]	20.32	30.8	20.0

where $I(s)$ and $I(t)$ are color values of adjacent pixels s and t. We estimate the color weight with an only pixel-wise color difference instead of normalizing with a whole frame factor, unlike HBP [30]. Our proposed approach simplifies the original process and is suitable for our on-the-fly procedure. The divider is implemented with a shifter. The smoothness term with color weighted becomes

$$E_s(l, l') = \delta \times \min_{l' \in L}(\lambda \left| l - l' \right|, \lambda T). \tag{19}$$

IV. EXPERIMENTAL RESULTS

In this section, we analyze the quality effect of each different block under consideration. The proposed memory passing architecture does not affect the performance and message update block. We implemented the proposed architecture and counterpart designs in TSMC 28nm technology. All synthesis results are under the same conditions, the "typical" operation condition. We use Design Compiler and PrimeTime-PX (Synopsys) to synthesize and analyze power. All SRAM modules in implementations and experiments are low power single port SRAMs in TSMC 28nm technology.

A. Qualitative Comparison

We evaluate the performance under two widely adopted datasets, Middlebury 2014 [36] and KITTI 2015 [37]. We adopt the absolute difference and Census (AD-Census) as our data term, E_d. To compare fairly, we implement TBP [18] and EBP [19] and use the same configuration. The first one is the Middlebury 2014 dataset [36] which is composed of indoor scenes. Results are shown in Fig. 13. Compared to other BP-based architectures, our proposed architecture provides smoother results and better edge-preserving. The second one is the KITTI 2015 dataset [37] which is composed of outdoor scenes. Results are shown in Fig. 14. In these results, our proposed architecture and EBP obtain smooth results in flat areas. Our proposed architecture provides incorrect propagation around the traffic sign in the lower right since added color weights.

Error rates for different datasets are shown in Table II. TBP [18] sacrifices quality for computation efficiency by setting an extremely small untruncated factor, T. Compared to EBP [19], the proposed architecture preserves better edge information and provides higher accuracy.

B. Memory-Efficient Message Passing

The computing resource and memory access comparison between the proposed architecture and its counterpart are

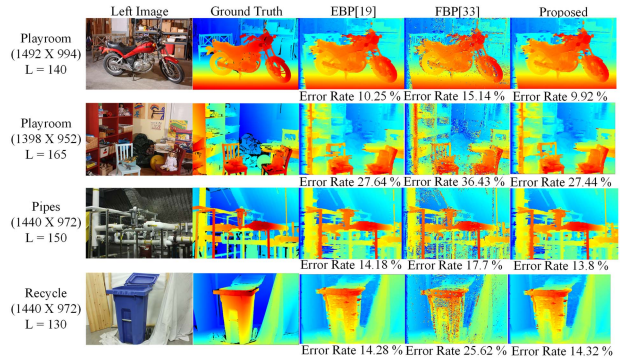


Fig. 13. **Results of various BP structures.** From left to right: left view images, ground truth depth maps, result of EBP, results of TBP and results of the proposed one. The error threshold for all testing image is 2.

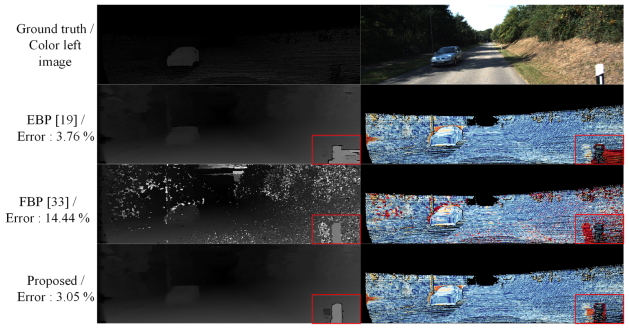


Fig. 14. **Testing image pairs for the proposed system of KITTI dataset frame 108.** The first row is EBP, the second row is TBP, and the third row is the proposed method.

TABLE III

THE COMPUTING RESOURCE AND MEMORY ACCESS COMPARISON (K IS THE NUMBER OF PE AND L IS THE DISPARITY RANGE). MEMP IS SHORT FOR MEMORY-EFFICIENT MESSAGE PASSING DATA FLOW [34]

Architecture	TBP [18]	MEMP [34]	Proposed
Adders	$3 * L$	$3 * L$(Backward) $2 * L$(Forward)	$3 * L$(Backward) $2 * L$(Forward)
SRAM	4 Block Buffers	2 Block Buffers	1 Block Buffer + K Line Buffers
Mem. read	4	3(Backward) 2(Forward)	3(Backward) 2(Forward)
Mem. write	1	1	1

shown in Table III. The number of adders is reduced by 20%, which is the reason for the nearly 20% improvement in software simulation [34].

The memory requirement of the on-chip memory in different structures is shown in Fig.15. MEMP [34] attains more than 40% SRAM reduction by grouping two messages into one. The proposed data path can further reduce the SRAM size with a block message buffer and K line buffers where K is the number of PEs. When L increases, the SRAM reduction by the proposed data flow increases as well. The design can attain 67.8% SRAM reduction when $L = 512$.

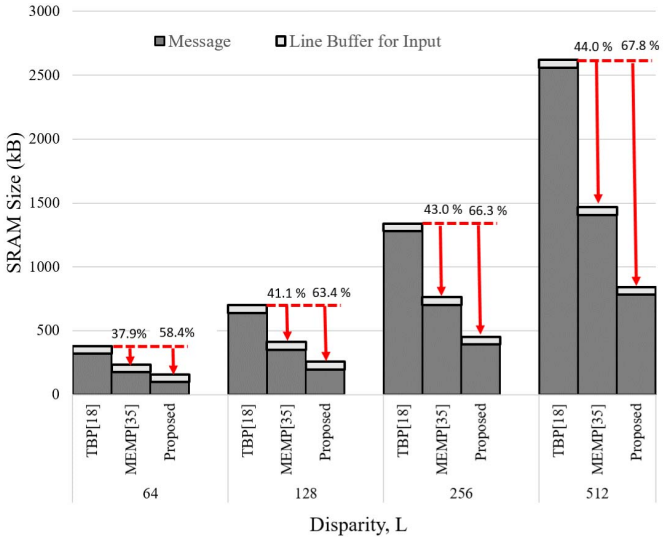


Fig. 15. SRAM size comparison between TBP [18], MEMP [34] and the proposed one. Please note that EBP [19] and LFBP [15] have the same size with TBP [18].

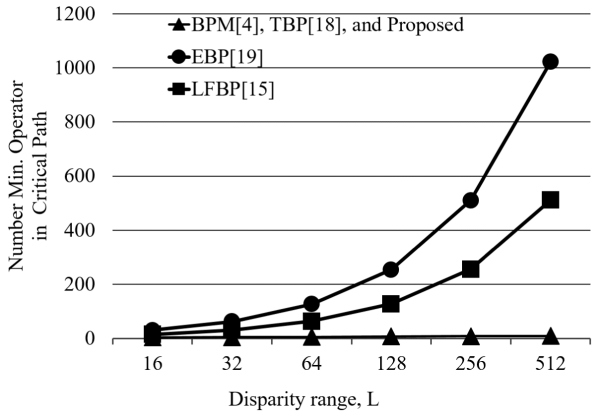


Fig. 16. Number of minimum operations in critical path with different architectures for various label counts (L) and $T = \frac{L}{8}$.

C. Fast Memory Update

Table I shows an analysis comparison between prior hardware architectures and the proposed one. The short delay characteristic of tree-based methods BPM [4] and TBP [18] is retained in the proposed architecture. Compared to path-based methods EBP [19] and LFBP [15], tree-based architectures are more suitable for large disparity range scenarios due to their shorter delay. Furthermore, the proposed efficient structure reduces the complexity from $O(L^2)$ to $O(Llog_2L)$ by eliminating redundancies. Critical paths are proportional to the disparity range in architecture EBP [19] and LFBP [15] both of which are not suitable for BP hardware implementation with large disparity ranges.

Minimum operators dominate the area of the BP processing element. Path-based architectures [15], [19] require fewer operators but are not suitable for large disparity ranges due to their long critical paths. Among tree-based methods, the proposed architecture requires fewer operators reflecting its gate count reduction.

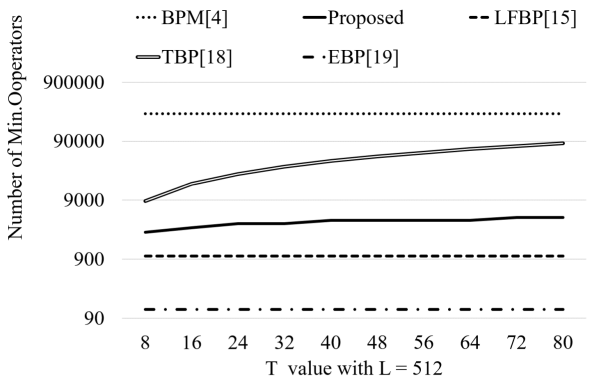


Fig. 17. Minimum operations comparison for different architectures with various T values (the label count is 512).

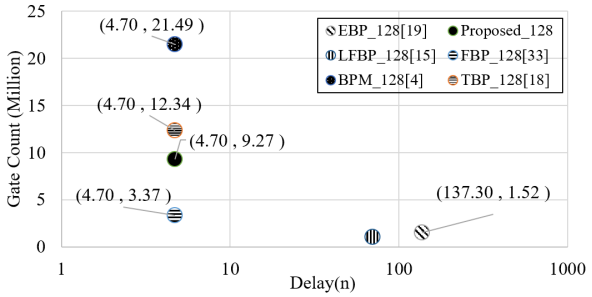


Fig. 18. **Synthesis results of various message update PE architectures at** $L = 128$**.** The format of the data label is (delay, gate count).

To illustrate the critical path issue, the number of minimum operators on the critical path for each architecture is shown in Fig. 16. Unlike tree-based implementations [4] [18] and the proposed one, the operators in the critical path increase proportionally with increasing disparity range. The critical paths of tree-based architectures are dominated by the global tree which has $log_2 L$ operators in the critical path. Hence, lines representing tree-based architectures overlap in Fig. 16.

We adopt a time-area term to demonstrate the hardware efficiency of each architecture. The time-area term is used to illustrate the trade-off between different architectures, where the term is defined as time-area term = delay × gate count.

Numbers of minimum operators in different hardware architectures with various T values are analyzed as shown in Fig. 17 whose disparity range is 512. When $T > 1$, the proposed architecture requires fewer minimum operators than the architecture in [18]. Observe that implementations EBP and LFBP require fewer minimum operators but have long critical paths, which are not friendly to large disparity range conditions. Among these architectures, only the proposed one and the architectures in [18] and [33] are affected by a different truncation parameter, T. As the proposed architecture avoids the redundancies found in TBP, it always has fewer operators, except for $T = 1$. When $T = 1$, the proposed architecture requires the same operators as FBP, which suggests that FBP can be considered as a special case of our proposed architecture with $T = 1$. FBP indicates a lossy architecture with a decrease in performance and TBP indicates the lossless one.

The synthesis results for each architecture with a disparity range of 128 are shown in Fig. 18. The critical paths of each hardware implementation are shown in the figure too.

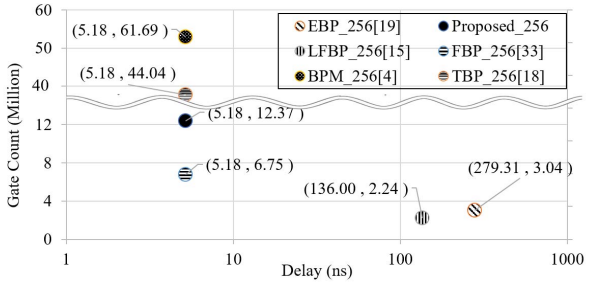


Fig. 19. **Synthesis results of various message update PE architecture at** $L = 256$**.** The format of the data label is (delay, gate count).

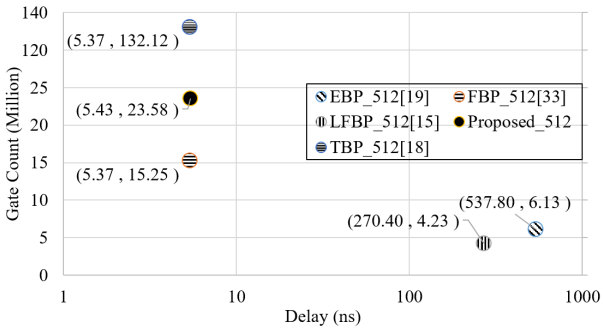


Fig. 20. **Synthesis results of various message update PE architecture at** $L = 512$**.** The format of the data label is (delay, gate count).

TABLE IV

SYNTHESIS REPORT OF THE BP PROCESSING ELEMENT FOR THE CONDITIONS L = 256 AND T = 32

	EBP [19]	TBP [18]	LFBP [15]	Ours
Gate Counts(M)	3.04	44.04	**2.24**	12.37
Compare to TBP [18] (%)	6.9	100	**5.0**	28.1
Critical Path(ns)	279.31	**5.18**	136.05	**5.18**
Compare to TBP [18] (%)	5392.1	**100**	2626.4	**100**
Time-area term	849.1	228.13	304.8	**64.1**
Reduction(%)	−272.2	0	−33.6	**71.9**

TABLE V

SYNTHESIS REPORT OF THE BP PROCESSING ELEMENT UNDER L = 512 AND T = 64 CONDITION

	EBP [19]	TBP [18]	LFBP [15]	Ours
Perf. Loss	**No**	**No**	Yes	**No**
Gate Counts(M)	6.13	132.12	**4.23**	23.58
Compare to TBP [18] (%)	4.6	100	**3.2**	17.8
Critical Path(ns)	537.8	**5.37**	270.4	**5.37**
Compare to TBP [18] (%)	1232	**100**	616	**100**
Time-area term	3296.7	709.5	1143.8	**126.6**
Reduction(%)	−364.7	0	−61.2	**82.2**

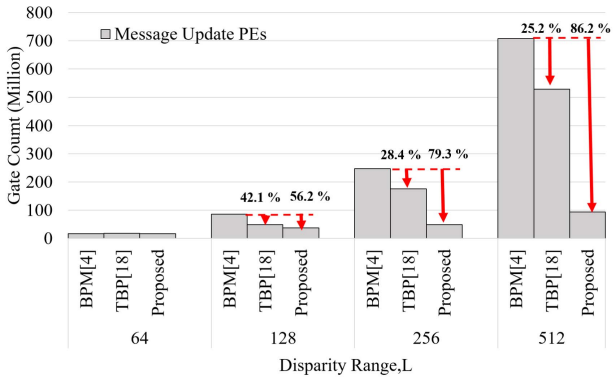


Fig. 21. **Gate counts comparison between each architecture.** Each architecture contains four message update PEs and corresponding cost computation PEs. Note that the advantage of our architecture becomes more obvious when L becomes larger. The gate count of BPM with 512 disparity range is simulated.

The synthesis results for each architecture with a disparity range of 256 and 512 are shown in Fig. 19 and Fig. 20, respectively. Please note that even though the message update PEs of LFBP and EBP exhibit larger delays, they have fewer gate counts. As shown in Fig. 20, there are no data for the implementation of BPM with a disparity range of 512, because the synthesis process could not converge within a week.

A comparison in time-area terms for various architectures are shown in Table IV; as can be seen, the proposed architecture reduces chip area by 71.9%, as compared to the lossless TBP with $L = 256$. It is worth mentioning that the reduction in minimum operators is approximately equal to the chip area reduction, and the reduction ratio increases with increasing disparity range. In Table V, an 82.2% time-area reduction is achieved. As to the long delay issue, path-based architectures have worse time-area term performances. When the disparity range is larger than 64, the proposed architecture can almost achieve the reductions achieved by the LFBP and TBP implementations, which are state-of-the-art implementations of path-based and tree-based architectures. Results show that the proposed structure attains a better trade-off between area and critical path when the disparity is larger than 64. When the disparity range increases, the more obvious the trade-off advantage of our proposed architecture becomes. A comparison of the gate counts for tree-based architectures is depicted in Fig. 21. It is evident that the proposed architecture offers a greater advantage when the disparity range increases. The gate count of BPM with a 512 disparity range is simulated since the synthesis process could not converge in a week.

D. Proposed System

The power required for a different number of depth candidates is shown in Fig. 22. We implemented different architectures of BP engines using identical synthesis conditions and SRAM modules. For a fair comparison, we simulated the SGM [12] engine and NG-fSGM [13] using our message update module and SRAM modules, because the original

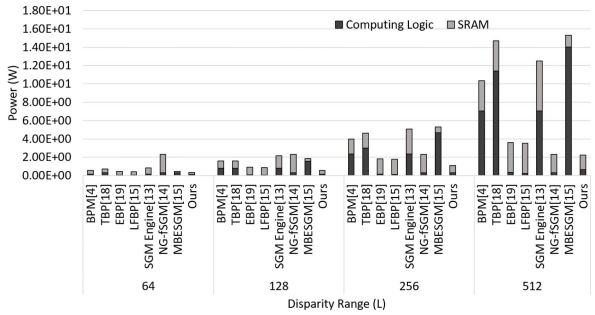


Fig. 22. **Relation between the power of each element and the number of depth candidates with different architectures.**

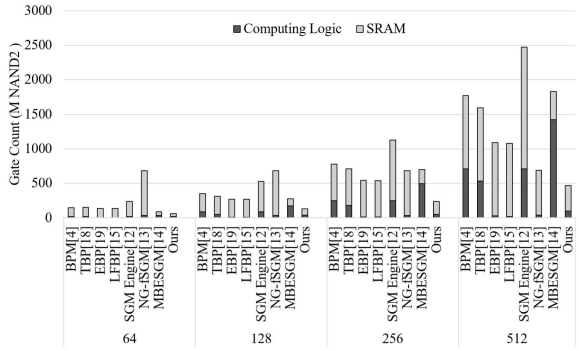


Fig. 23. **Relation between the gate count of each element and the number of depth candidates with different architectures.**

disparity range and implementation technology of these architectures differ from ours. The adopted message update module for the SGM engine [12] is identical to that of BPM [38], while the SRAM size is linearly scaled with the disparity range. The adopted message update module for the NG-fSGM [13] is that of BPM with a disparity range of 64 because the processing labels are fixed at each iteration and the SRAM size is fixed at 1568 kB for each disparity range. According to [12] [13], four and eight PEs are used for the SGM engine and NG-fSGM, respectively. We also simulated MBESGM according to a previous report [14] by using eight PEs, similar to NG-fSGM. The SRAM sizes of the SGM engine [12], NG-fSGM [13], and MBESGM [14] are simulated based on their reports. Although MBESGM requires a smaller SRAM size, its computational complexity increases with the disparity range; thus, the power consumption is dominated by computing logic.

Gate counts for different numbers of depth candidates are shown in Fig. 23. Based on previous power analyses, data of the SGM [12] engine,NG-fSGM [13], and MBESGM [14] are simulated under identical synthesis conditions. Although EBP [19] and LPBP [15] exhibit lower computational complexity, the gate counts of these path-based architectures are dominated by SRAM.

We compare five features to determine the most suitable architecture for large disparity ranges in a radar chart shown in Fig. 24. These features are accuracy, speed, hardware complexity, SRAM size, and hardware efficiency. BPM [4]

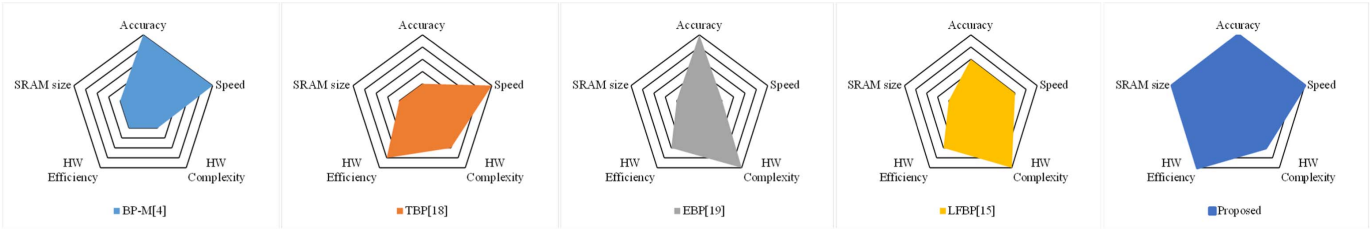


Fig. 24. **Comparison between various architecture.** These radar charts plot accuracy, speed, SRAM size, hardware complexity and hardware-efficiency. From the left to the right are BP-M [4], EBP [19], TBP [18], LFBP [15], and the proposed architecture.

TABLE VI

COMPARISON WITH DIFFERENT CONFIGURATION WITH L = 512. THE GATE COUNT OF COST COMPUTATION UNITS IS 3.90 MILLION

Message Passing	Message Update	SRAM Size (KB)	Gate Count (M)	Latency(ns)
TBP [18]		2560 + 60	528.48 + 3.90	5.38
EBP [19]		2560 + 60	6.13 + 3.90	537.8
LFBP [15]		2560 + 60	16.92 + 3.90	270.4
TBP [18]	Proposed	2560 + 60	94.32 + 3.90	5.38
Proposed	TBP [18]	784 + 60	528.48 + 3.90	5.38
Proposed		784 + 60	94.32 + 3.90	5.38

and TBP [18] are less hardware efficient but faster, and this is one of the most important features in real-time application. Although EBP [19] and LFBP [15] are much slower but more hardware efficient, their speed reduces their advantage for large disparity cases. Our architecture, however, exhibits all three features, i.e. it is faster, more hardware efficient, and more accurate. The comparison between previous architectures and the proposed architecture is listed in Table VI.

V. DISCUSSION

In this study, we concentrate on enhancing the hardware and memory-efficiency of stereo matching. We adopt the energy functions of TBP [18] and LFBP [15]. A weighted message [30] preserves discontinuities that are over-smoothed by global methods. Parameters in the proposed architecture are fixed; however, adaptive or learned parameters that consider the input characters can further improve the depth quality.

Moreover, compact energy representations [29] [39] [40] can further enhance the memory-efficiency of MRF algorithms. Most studies focus on improving the accuracies of their reconstructed energy functions. Slow and cost expensive reconstruction processes are crucial problems for real-time applications and hardware design. A fast and cost-effective architect can enable these energy compression techniques to be more practical.

VI. CONCLUSION

A hardware-efficient BP-based architecture is proposed. It features a memory-efficient data flow and a hardware-efficient message updating processing element: both features are beneficial in large disparity stereo matching scenarios. An integrated system is proposed to demonstrate that this design is more hardware efficient and more suitable for large disparity range scenarios than existing systems. The current state-of-the-art implementations are not practical for disparity ranges larger than 256 because of the large on-chip memory requirements, as well as delay issues. To the best of our knowledge, the proposed system is the first stereo matching engine that can support full-HD resolution input images and a disparity range of 512 with less than 1 MB SRAM. Compared to state-of-the-art implementations, the proposed implementation offers a 67.8% SRAM reduction and an 86.2% gate count reduction of message update units at a disparity range of 512—all this without a reduction in performance, and this is the most computation-intensive aspect of a BP engine. In the future, higher hardware efficiency could be achieved via advanced loss energy function compression techniques with a cost-effective architecture.

REFERENCES

[1] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *Int. J. Comput. Vis.*, vol. 47, nos. 1–3, pp. 7–42, Apr. 2002.

[2] K. Zhang, J. Lu, and G. Lafruit, "Cross-based local stereo matching using orthogonal integral images," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 19, no. 7, pp. 1073–1079, Jul. 2009.

[3] K.-J. Yoon and I. S. Kweon, "Adaptive support-weight approach for correspondence search," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 4, pp. 650–656, Apr. 2006.

[4] 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.

[5] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 11, pp. 1222–1239, Nov. 2001.

[6] V. Kolmogorov, "Convergent tree-reweighted message passing for energy minimization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 10, pp. 1568–1583, Oct. 2006.

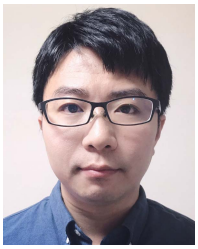
[7] Q. Yang, "A non-local cost aggregation method for stereo matching," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 1402–1409.

[8] I. Ernst and H. Hirschmüller, "Mutual information based semi-global stereo matching on the GPU," in *Proc. Int. Symp. Vis. Comput.* Berlin, Germany: Springer, 2008.

[9] X. Cheng, P. Wang, and R. Yang, "Depth estimation via affinity learned with convolutional spatial propagation network," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 103–119.

[10] J.-R. Chang and Y.-S. Chen, "Pyramid stereo matching network," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 5410–5418.

[11] A. Suleiman, Y.-H. Chen, J. Emer, and V. Sze, "Towards closing the energy gap between HOG and CNN features for embedded vision," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2017, pp. 1–4.
[12] M. Roszkowski, "Optimization of semi-global stereo matching for hardware module implementation," *Proc. SPIE*, vol. 9290, Nov. 2014, Art. no. 92902T.
[13] Z. Li, J. Xiang, L. Gong, D. Blaauw, C. Chakrabarti, and H. S. Kim, "Low complexity, hardware-efficient neighbor-guided SGM optical flow for low-power mobile vision applications," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 29, no. 7, pp. 2191–2204, Jul. 2019.
[14] X. Zhang, H. Dai, H. Sun, and N. Zheng, "Algorithm and VLSI architecture co-design on efficient semi-global stereo matching," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 11, pp. 4390–4403, Nov. 2020.
[15] H. Chen, C. Huang, S. Wu, C. Hung, T. Ma, and L. Chen, "A 1920×1080 30 fps 611 mw five-view depth-estimation processor for light-field applications," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2015, pp. 1–3.
[16] Y.-C. Tseng and T.-S. Chang, "Architecture design of belief propagation for real-time disparity estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 20, no. 11, pp. 1555–1564, Nov. 2010.
[17] T. Yu, R.-S. Lin, B. Super, and B. Tang, "Efficient message representations for belief propagation," in *Proc. IEEE 11th Int. Conf. Comput. Vis.*, Oct. 2007, pp. 1–8.
[18] C.-K. Liang, C.-C. Cheng, Y.-C. Lai, L.-G. Chen, and H. H. Chen, "Hardware-efficient belief propagation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 21, no. 5, pp. 525–537, May 2011.
[19] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient belief propagation for early vision," *Int. J. Comput. Vis.*, vol. 70, no. 1, pp. 41–54, Oct. 2006.
[20] A. A. Sharma, K. Neelathalli, D. Marculescu, and E. Nurvitadhi, "Hardware-efficient stereo estimation using a residual-based approach," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, May 2013, pp. 2693–2696.
[21] Q. Yang, L. Wang, R. Yang, S. Wang, M. Liao, and D. Nister, "Real-time global stereo matching using hierarchical belief propagation," in *Proc. Brit. Mach. Vis. Conf. (BMVC)*, vol. 6, 2006, pp. 989–998.
[22] J. Choi and R. A. Rutenbar, "Video-rate stereo matching using Markov random field TRW-S inference on a hybrid CPU+FPGA computing platform," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 2, pp. 385–398, Feb. 2016.
[23] W. Wang, J. Yan, N. Xu, Y. Wang, and F.-H. Hsu, "Real-time high-quality stereo vision system in FPGA," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 25, no. 10, pp. 1696–1708, Oct. 2015.
[24] Z. Li *et al.*, "A 1920 × 1080 30-frames/s 2.3 TOPS/W stereo-depth processor for energy-efficient autonomous navigation of micro aerial vehicles," *IEEE J. Solid-State Circuits*, vol. 53, no. 1, pp. 76–90, Jan. 2018.
[25] F. Besse, C. Rother, A. Fitzgibbon, and J. Kautz, "PMBP: PatchMatch belief propagation for correspondence field estimation," *Int. J. Comput. Vis.*, vol. 110, no. 1, pp. 2–13, Oct. 2014.
[26] Y. Li, D. Min, M. S. Brown, M. N. Do, and J. Lu, "SPM-BP: Sped-up PatchMatch belief propagation for continuous MRFs," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 4006–4014.
[27] M. P. Heinrich, I. J. A. Simpson, B. W. Papieü, S. M. Brady, and J. A. Schnabel, "Deformable image registration by combining uncertainty estimates from supervoxel belief propagation," *Med. Image Anal.*, vol. 27, pp. 57–71, Jan. 2016.
[28] D. J. Crandall, A. Owens, N. Snavely, and D. P. Huttenlocher, "SfM with MRFs: Discrete-continuous optimization for large-scale structure from motion," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 12, pp. 2841–2853, Dec. 2013.
[29] Q. Yang, L. Wang, and N. Ahuja, "A constant-space belief propagation algorithm for stereo matching," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2010, pp. 1458–1465.
[30] Q. Yang, L. Wang, R. Yang, H. Stewenius, and D. Nister, "Stereo matching with color-weighted correlation, hierarchical belief propagation, and occlusion handling," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 3, pp. 492–504, Mar. 2009.
[31] A. Ahmadzadeh, H. Madani, K. Jafari, F. S. Jazi, S. Daneshpajouh, and S. Gorgin, "Fast and adaptive bp-based multi-core implementation for stereo matching," in *Proc. 11th IEEE/ACM Int. Conf. Formal Methods Models Codesign*, Dec. 2013, pp. 135–138.
[32] C.-C. Cheng, C.-T. Li, C.-K. Liang, Y.-C. Lai, and L.-G. Chen, "Architecture design of stereo matching using belief propagation," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2010, pp. 4109–4112.
[33] 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 Int. Conf. Acoust., Speech Signal Process.*, Apr. 2009, pp. 745–748.
[34] S.-S. Wu, H.-H. Chen, C.-H. Tsai, and L.-G. Chen, "Memory efficient architecture for belief propagation based disparity estimation," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2015, pp. 2521–2524.
[35] S.-S. Wu, C.-H. Tsai, and L.-G. Chen, "Efficient hardware architecture for large disparity range stereo matching based on belief propagation," in *Proc. IEEE Int. Workshop Signal Process. Syst. (SiPS)*, Oct. 2016, pp. 236–241.
[36] D. Scharstein *et al.*, "High-resolution stereo datasets with subpixel-accurate ground truth," in *Proc. German Conf. Pattern Recognit.* Cham, Switzerland: Springer, 2014.
[37] M. Menze and A. Geiger, "Object scene flow for autonomous vehicles," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 3061–3070.
[38] R. Szeliski *et al.*, "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.
[39] T. Mollenhoff, E. Laude, M. Moeller, J. Lellmann, and D. Cremers, "Sublabel–accurate relaxation of nonconvex energies," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 3948–3956.
[40] T. Mollenhoff and D. Cremers, "Sublabel-accurate discretization of non-convex free-discontinuity problems," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 1192–1200.



Sih-Sian Wu (Member, IEEE) received the B.S. and M.S. degrees in electrical engineering from National Cheng Kung University, Tainan, Taiwan, in 2009 and 2012, respectively. From 2012 to 2013, he was a Research Assistant with the School of Electrical and Electronic Engineering, Nanyang Technology University, Singapore. He is currently pursuing the Ph.D. degree in electronic engineering with National Taiwan University, Taipei, Taiwan. His current research interests include 3D image processing, very large scale integration design, and chip design. In 2009, he received the Best Paper Award of International Computer Symposium (ICS). He is also a member of the honor society Phi Tan Phi.

Hon-Hui Chen (Member, IEEE) was born in Taipei, Taiwan, in 1978. He received the B.S., M.S., and Ph.D. degrees in electrical engineering from National Taiwan University, Taipei, Taiwan, in 2000, 2002, and 2015, respectively. His research interests include algorithm developments, system level analysis/design/integration, and system-on-a-chip (SoC) realization for bio-medical applications.

Liang-Gee Chen (Fellow, IEEE) was born in Yunlin, Taiwan, in 1956. He received the B.S., M.S., and Ph.D. degrees in electrical engineering from National Cheng Kung University in 1979, 1981, and 1986, respectively. He was an Instructor (1981–1986) and an Associate Professor (1986–1988) with the Department of Electrical Engineering, National Cheng Kung University. In the military service from 1987 to 1988, he was an Associate Professor with the Defense Management College, Institute of Resource Management. In 1988, he joined the Department of Electrical Engineering, National Taiwan University. From 1993 to 1994, he was a Visiting Consultant with the DSP Research Department, *AT&T* Bell Lab, Murray Hill. In 1997, he was a Visiting Scholar with the Department of Electrical Engineering, University of Washington, Seattle. He is currently a Professor with National Taiwan University. In 2004, he is also the Executive Vice President and the General Director of Electronics Research and Service Organization (ERSO), Industrial Technology Research Institute (ITRI). His current research interests include DSP architecture design, video processor design, and video coding systems. He is also a member of the honor society Phi Tan Phi. He received the Best Paper Award from ROC Computer Society in 1990 and 1994. From 1991 to 1999, he received the Long-Term (Acer) Paper Awards annually. In 1992, he received the Best Paper Award of the 1992 Asia–Pacific Conference on Circuits and Systems in VLSI design track. In 1993, he received the Annual Paper Award of Chinese Engineer Society. In 1996, he received the Out-Standing Research Award from NSC, and the Dragon Excellence Award for Acer. He was elected as the IEEE Circuits and Systems Distinguished Lecturer from 2001–2002. He was the General Chairman of the 7th VLSI Design CAD Symposium and the 1999 IEEE Workshop on Signal Processing Systems: Design and Implementation. He has been serving as an Associate Editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY since June 1996 and for the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION SYSTEMS since January 1999. He has been an Associate Editor of the *Circuits, Systems, and Signal Processing* journal since 1999. He served as a Guest Editor for *The Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology* in November 2001. He is also an Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS II: ANALOG AND DIGITAL SIGNAL PROCESSING. Since 2002, he has also been an Associate Editor of PROCEEDINGS OF THE IEEE.