High-speed parallel very large scale integration architecture for global stereo matching

Sungchan Park and Hong Jeong
POSTECH, Department of Electronic and Electrical Engineering, Pohang 790-784, Korea
E-mail: mrzoo@postech.ac.kr

Abstract. *Although stereo matching algorithms based on belief propagation (BP) tend to show excellent matching performance, their huge computational complexity has been the major barrier to real-time applications. In this light, we propose a parallel very large scale integration (VLSI) architecture for BP computation, which has only simple integer operations and shows low matching error rate for the Middlebury database.* © 2008 SPIE and IS&T.
[DOI: 10.1117/1.2892680]

1 Introduction

Stereo matching algorithms find corresponding points in a pair of images to locate 3-D positions. They can be classified into either local or global matching approaches.[1] Local approaches, like correlation and dynamic programming methods, deal only with subimages. These approaches have the advantage of real-time speed,[2,3] but tend to produce high errors. In contrast, the global approaches, like graph cuts and belief propagations (BPs),[4] deal with full images. These approaches have the advantage of low errors, but tend to execute huge computational loads. In real-time applications, like robot vision, the stereo matching system should be compact and fast. In this context, we present a very large scale integration (VLSI) architecture for stereo matching with BP.

2 Belief Propagation Formulation for Stereo Matching

Given the left and right images g^r, g^l, and the parameters c_d, c_v, K_d, and K_v, we describe the energy model for a 2-D Markov random field (MRF) as follows.[4]

$$\hat{d} = \arg\min_{\mathrm{d}} E(d), \quad E(d) = \sum_{\mathbf{p},\mathbf{q}\in N} V(d_\mathbf{p}, d_\mathbf{q}) + \sum_{\mathbf{p}\in P} D(d_\mathbf{p}),$$

$$D(d_\mathbf{p}) = \min[c_d | g^r(d_\mathbf{p} + \mathbf{p}) - g^r(d_\mathbf{p})|, K_d],$$

$$V(d_\mathbf{p}, d_\mathbf{q}) = \min(c_v | d_\mathbf{p} - d_\mathbf{q}|, K_v),$$

where $D(d_\mathbf{p})$ denotes the data cost of the label $d_\mathbf{p} \in [0, d_{\max}-1]$ at the pixel $\mathbf{p}$ in the image P, and $V(d_\mathbf{p}, d_\mathbf{q})$ denotes the discontinuity cost between the label $d_\mathbf{p}$ and $d_\mathbf{q}$ of the neighbor nodes N. The disparity $\hat{d}$ can be estimated using the BP's message_update_function $(\mathbf{p}, \mathbf{q}, k, k+1)$ and the decision_function $(\mathbf{p}, K)$ as follows:

$$m_{\mathbf{pq}}^{k+1}(d_\mathbf{q}) = \min_{d_\mathbf{p}} \left\{ V(d_\mathbf{p}, d_\mathbf{q}) + D_\mathbf{p}(d_\mathbf{p}) + \sum_{\mathbf{u}\in N(\mathbf{p})\backslash\mathbf{q}} [m_{\mathbf{up}}^k(d_\mathbf{p}) - \alpha] \right\}, \quad (1)$$

$$\hat{d}_\mathbf{p} = \arg\min_{d_\mathbf{p}} \left[D_\mathbf{p}(d_\mathbf{p}) + \sum_{\mathbf{q}\in N(\mathbf{p})} m_{\mathbf{qp}}^K(d_\mathbf{p}) \right], \quad \alpha = \sum_{d_\mathbf{p}} \frac{m_{\mathbf{sp}}^k(d_\mathbf{p})}{d_{\max}}, \quad (2)$$

where $N(\mathbf{p})\backslash\mathbf{q}$ denotes the neighbors of $\mathbf{p}$ other than $\mathbf{q}$, and α denotes the normalization value. At each node $\mathbf{p}$, the message $m_{\mathbf{pq}}^{\mathbf{k+1}}(d_\mathbf{q})$ at $k+1$ iteration times is updated synchronously using neighboring message $m_{\mathbf{up}}^k(d_\mathbf{p})$ and sent from node $\mathbf{p}$ to neighbor node $\mathbf{q}$. After K iterations, the $\hat{d}_\mathbf{p}$ at each node is decided by Eq. (2).

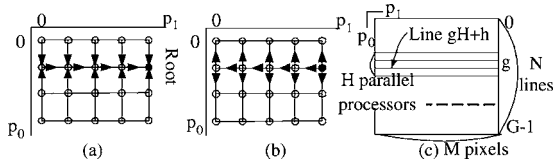

Fig. 1 Update sequence at k iteration times on 2-D MRF: (a) inward processing, (b) outward processing, and (c) parallel processing within group.

3 Proposed Architecture of Stereo Matching

Generally, BP can be separated into two methods, mainly according to the update style. The first method updates all the nodes synchronously on the 2-D MRF.[4] The second method updates all the nodes sequentially; first in the inward direction from the leaf to the root and next in the reverse direction.[5,6] The sequential method needs to update only once at each node and obtains the final results. Therefore, the nodes can be propagated fast with the small number of operations. In Ref. 5, the authors reported that the sequential update based on spanning trees in MRF can achieve fast convergence. We applied the tree structure to each scan line, as shown in Figs. 1(a) and 1(b). The tree messages are updated using messages from the neighboring scan lines that have been determined in the previous iteration times. For the image with $M \times N$ pixels in Fig. 1(c), N scan lines can be separated into G groups, and the H lines of each group $g \in [0, G-1]$ can be processed in parallel with H processors. This observation is shown in our VLSI parallel sequences as follows. A node located in a pixel is denoted by a 2-D vector $\mathbf{p}=[p_0\ p_1]^T$.

For synchronous iteration k from 1 to K,
  for group g from 0 to $G-1(=N/H-1)$,
    for each parallel processor h from 0 to $\mathbf{H}-1$, $(\mathbf{p}=[gH+hp_1]^T)$.

Paper 06080LR received May 23, 2006; revised manuscript received Oct. 6, 2007; accepted for publication Jan. 7, 2008; published online Mar. 13, 2008.
1017-9909/2008/17(1)/010501/3/$25.00 © 2008 SPIE and IS&T.

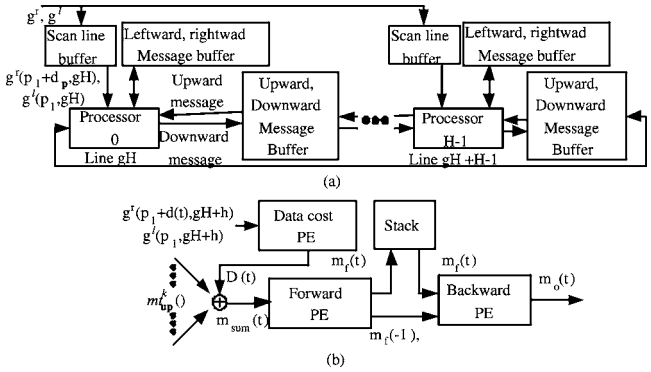

Fig. 2 Parallel and pipeline architecture: (a) processor array and (b) PE.

1. Inward processing to root, for $p_1=0,\ldots,M-1$,
   message_update_function $(\mathbf{p},\mathbf{p}+[0\ 1]^T,k,k)$ for rightward message.
2. Outward processing to leaf, for $p_1=M-1,\ldots,0$,
   message_update_function $(\mathbf{p},\mathbf{p}-[0\ 1]^T,k,k)$ for leftward message
   message_update_function $(\mathbf{p},\mathbf{p}+[1\ 0]^T,k,k+1)$ for downward message
   message_update_function $(\mathbf{p},\mathbf{p}-[1\ 0]^T,k,k+1)$ for upward message
   decision_function $(\mathbf{p},K)$.

As shown in Fig. 2(a), the H processors calculate the messages in the parallel, receiving the left and right pixel data from the H scan line buffers, and reading and writing with each message buffer. The processor consists of the processing elements (PE) PE^f,PE^b,PE^u, PE^d, and PE^o. Using the image data and the neighboring messages, PE^f calculates the forward message in the inward time and PE^b, PE^u, PE^d, and PE^o calculate each direction's message and disparity $\hat{d}$ in the outward time.

4 Architecture of Processing Element

The PE is the basis logic to calculate the new message at each node as follows.

$$m_{\mathbf{pq}}^{k+1}(d_{\mathbf{p}}) = \min_{d_{\mathbf{q}}\in[0,d_{\max}-1]} V(d_{\mathbf{p}},d_{\mathbf{q}}) + m_{\text{sum}}(d_{\mathbf{q}}) - \alpha,$$

$$m_{\text{sum}}(d_{\mathbf{q}}) = D_{\mathbf{p}}(d_{\mathbf{q}}) + \sum_{\mathbf{u}\in N(\mathbf{p})\backslash\mathbf{q}} m_{\mathbf{up}}^{k}(d_{\mathbf{q}}).$$

When $V(t,l)=\min(C_v|t-l|,K_v)$, by the recursive backward and forward methods of the distance transform,[4] the time complexity is $O(5D)$ for D disparity levels. Due to our pipeline structure, 2-D clocks are necessary for calculating the message $m_{\mathbf{pq}}^{k+1}(d_{\mathbf{p}})$. In the forward initialization, $D_1(-1)=B$, $D_2(-1)=B$ (B is as big as possible).

For clock t from 0 to $D-1$ in the forward PE,

$$D_1(t) = \min[m_{\text{sum}}(t), D_1(t-1) + C_v],$$

$$D_2(t) = \min[m_{\text{sum}}(t), D_2(t-1)],$$

$$m_f(t) = D_1(t), \quad m_f(-1) = D_2(D-1) + K_v, \quad \alpha = D_2(D-1). \tag{3}$$

In the backward initialization, $D_3(-1)=B$.

For clock t from 0 to $D-1$ in the backward PE,

$$D_3(t) = \min(m_f(D-1-t), D_3(t-1) + C_v),$$

$$m_{\mathbf{pq}}^{k+1}(t) = \min[D_3(t), m_f(-1)] - \alpha.$$

Figure 2(b) shows the PE architecture. The data cost PE calculates the data cost $D(t)$ from the left and right image pixels. The forward PE reads $m_{\text{sum}}(t)$, which is the sum of the messages and the data cost; outputs the forward cost $m_f(t)$, which is the minimum value between $m_{\text{sum}}(t)$ and $D_1(t-1)+C_v$; and saves it to the stack. In Eq. (3), the parameters are calculated for the backward time. In our system, the minimum cost of $m_{\text{sum}}(t)$ is used for the normalized parameter α.

The backward processor reads the $m_f(D-1-t)$ from the stack, calculates the minimum value $D_3(t)$ recursively, outputs the minimum value between $D_3(t)$ and the parameter $m_f(-1)$, and then subtracts it by α for the normalization.

5 Experimental Results

As shown in Fig. 3, we tested our system using four gray-scale images and ground truth data from the Middlebury

Table 1 Error rate comparison in several images.

Methods (iteration)	Tsukuba 384×288	Venus 436×383	Map 436×380	Sawtooth 284×216	Speed performance
Local case[a]	4.3%	1.5%	0.8%	1.3%	No real-time hardware
Local case[b]	4.3%	2.2%	0.8%	2.1%	SIMD in Pentium 4, 320×240, 7 fr/s
BP	4.8%	8.9%	4.2%	1.4%	No real-time hardware
Our chip(12)	2.6%	0.8%	0.2%	0.8%	FPGA 256×240, 25 fr/s

[a]Reference 2.
[b]Reference 3.

(a) (b) (c) (d)

Fig. 3 Left images: (a) Tsukuba, (b) Venus, (c) Sawtooth, and (d) Map.

database.[1] The error rate represents the percentage of disparity error of more than 1 between output $d(x,y)$ and ground truth $d_{\mathrm{TRUE}}(x,y)$,

$$\mathrm{error}(\%) = \frac{100}{N_m} \sum_{(x,y)\in P_m} [|d(x,y) - d_{\mathrm{TRUE}}(x,y)| > 1],$$

$$N_m = \sum_{(x,y)\in P_m} 1,$$

where P_m is the pixel area except for the occlusion part, and N_m is the pixel number in this area. As shown in Fig. 4 and Table 1, our system, iterated a small number of times, shows the results superior to the local method[2,3] and BP in the Tsukuba image. Based on Fig. 4(b), our disparity error converges rapidly around 12 iterations.

Given an $M \times N$ image, D disparity levels, and T iterations, we need only 2-D forward and backward processing clocks in PE, and $2M$ inward and outward processing steps at each scan line. $G(=N/H)$ groups are iterated T times by H processors in parallel at F frame rates. From Fig. 5(b), the total necessary number of clocks to process F frames in one second is $49M$ clocks $[=2\mathrm{D}(2M)(N/H)TF=(16\times 2)\times(256\times 2)\times(240/24)\times 12\times 25]$. Our system's 65-MHz clock was enough for real-time processing.

As shown in Fig. 2(a), the overall memory is spent for the scan line buffer and message buffer. For real-time processing, our processors should be allowed to access a pair of images in the scan line buffer, while the buffer loads new images from the cameras continuously. Hence, the size of

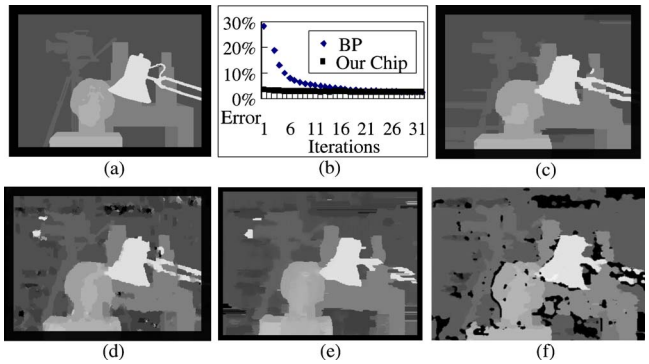

Fig. 4 Comparison of outputs in Tsukuba: (a) ground truth, (b) convergence rate, (c) our system at 12 iterations, (d) BP at 12 iterations, (e) local method,[2] and (f) local method.[3]

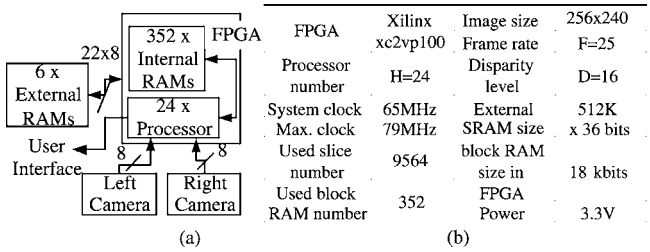

Fig. 5 Chip specifications: (a) overall system and (b) resource usage and output performance.

buffer for four images is allocated on the field programmable gate array (FPGA), which means 2 Mbits[=4 $\times(256\times 240\times 8)$]. Given $C(=4)$ bits as the size of message at each disparity level and $H(=24)$ processors, the message buffer memory size is as follows:

leftward message: 1.5 kbits=HCD,

rightward message: 0.4 Mbits=HCDM,

upward and downward message: 8 Mbits $=2HCDMG$.

In outward processing, the newly calculated leftward message is only used for the next pixel processing. One scan line size of rightward messages in inward time should be stored for outward processing. Since the upward and downward messages are updated synchronously for the entire image, we need to store all the pixel messages in the image. Due to this big memory size, 22 processors access the external memory through an 8×22-bit data bus. Two processors use the internal block RAMs on the FPGA. The overall memory resource usage is described in Fig. 5.

6 Conclusions

Although BP produces good error performances in the area of image processing, the VLSI architecture has not been fully studied yet. In this context, we propose a parallel VLSI architecture for stereo matching. The test system has only 16 disparity levels, which might not be satisfactory for 3-D recognition tasks. However, for applications, like real-time Z-keying and target-background separation, where low disparity error at the object boundary is important, the proposed chip can be effectively used.

References

1. D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *Int. J. Comput. Vis.* **47**(1/2/3), 7–42 (Apr. 2002).
2. H. Hirschmuller, "Improvements in real-time correlation-based stereo vision," *IEEE Workshop Stereo Multi-Baseline Vision*, pp. 141–148 (2001).
3. Y. Sukjune, P. Sung-Lee, K. Sungchul, and K. Yoon-Keun, "Fast correlation-based stereo matching with the reduction of systematic errors," *Pattern Recogn. Lett.* **26**(14), 2221–2231 (2005).
4. P. F. Felzenszwalb and D. R. Huttenlocher, "Efficient belief propagation for early vision," *Proc. 2004 IEEE CVPR* 1, I-261–I-268 (2004).
5. J. W. Martin, M J. Wainwright, T. S. Jaakkola, and A. S. Willsky, "Tree-based reparameterization framework for analysis of sum-product and related algorithms," *IEEE Trans. Inf. Theory* **49**(5), 1120–1146 (2003).
6. M. F. Tappen and W. T. Freeman, "Comparison of graph cuts with belief propagation for stereo, using identical MRF parameters," in ICCV, pp. 900–907 (2003).