FAST BELIEF PROPAGATION PROCESS ELEMENT FOR HIGH-QUALITY STEREO ESTIMATION

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

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

ABSTRACT

Belief propagation is a popular global optimization technique for many computer vision problems. However, it requires extensive computation due to the iterative message passing operations. In this paper, we present a new process element (PE) for efficient message construction. The efficiency is gained by exploiting the unique characteristics of the generalized Potts model (truncated linear mode) of the smoothness term in the Markov random field. For stereo estimation with L disparity values, the algorithm successfully reduces the computation from $\mathcal{O}(L^2)$ to $\mathcal{O}(L)$ and retains the high throughput and low latency. Compared with the direct message construction PE, our method achieves 87.14% computation saving and a 94.38% PE area reduction.

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

1. INTRODUCTION

Many problems in computer visions and image processing attempt to assign an optimal label to each node (pixel, block, or other element) in the image. A label stands for a local quantity, which can be a disparity vector or a motion vector.

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

$$\{l_p\} = \arg\min_{\{l_p\}\in L\times L\times L\ldots\times L} \left\{ \sum_{p\in P} E_d(l_p) + \sum_{(p,q)\in G} E_s(l_p, l_q) \right\}, \quad (1)$$

where L is the set of possible labels, P is the set of all nodes and G is the specified neighborhood, such as the 4-nearest neighbors.

Recently, efficient algorithms, such as graph cuts [1], (loopy) belief propagation [2], and numerous variants [3], for solving Eq. (1) have received increasing attention. They can find a strong locally optimal solution in polynomial time and thus enable many applications such as disparity estimation [6], image denoising, inpainting, image stitching, bi-layer segmentation, etc [3].

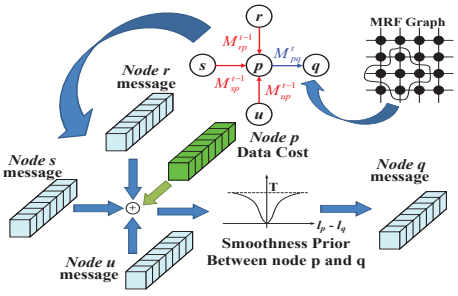

Fig. 1. Message computation of belief propagation

While the software implementation of these algorithms works well, the hardware implementation has not been fully addressed in the past. The hardware implementation requires different considerations: bandwidth, internal and external memory size, degree of parallelism and regularity of memory access, etc.

In our initial study, we found that belief propagation among others has the highest potential for hardware implementation [5]. It is highly parallel and only uses simple operations. We have reduced the bandwidth and memory overhead for hardware implementation in [4]. However, the straightforward hardware implementation is still prohibitive due to the amount of computation for message construction.

In this paper, we focus on the computational issue and propose a new processing element (PE) for the message construction, particularly for stereo estimation applications. The algorithm behind the PE exploits the unique property of the smoothness terms commonly used in stereo estimation. Compared with the PE that performs the traditional message construction, our PE requires only a fraction of the computation and area.

The rest of the paper is organized as follows. In Section 2, we review the original belief propagation technique and analyze its message computation. In Section 3, we present the proposed hardware-oriented fast message computation

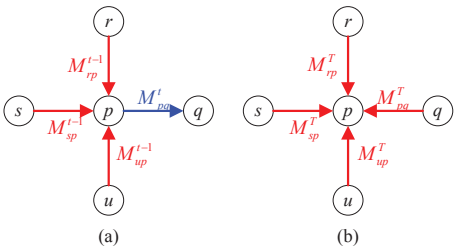

Fig. 2. (a) A message at iteration t from p to q is constructed using the messages from r, s, and u to p at iteration $t-1$. (b) The node p collects all messages from the neighbors to decide the best label.

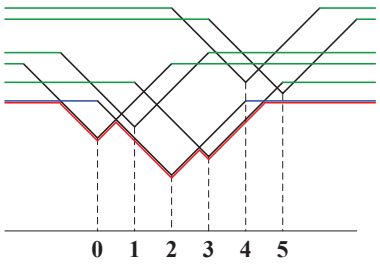

Fig. 3. An illustration of the truncated linear envelope of 6 cones in the case of one-dimensional labels. The red line is the final message. λ=2 and T=10.

algorithm and the coresponding PE design in detail. In Section 4, the comparison between the proposed algorithm and previous ones from the hardware perspective is given. Finally, we conclude this paper in Section 5.

2. BELIEF PROPAGATION

Belief propagation (BP) iteratively performs message passing operations. At iteration t, each node p sends a $|L|$-dimensional message M_{pq}^t to its neighbor q. Each entity $M_{pq}^t(l)$ in the message can be expressed as follows:

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

where L is the set of all labels, $|L|$ is the number of labels, and $\mathbf{N}_p$ is the set of the neighbors of p, as shown in Fig. 1. $M_{pq}^t(l_q)$ encodes the opinion of p about assigning label l_q to q. Node p first scans all labels l' and decides the one that gives the greatest support for assigning l to q based on 1) the smoothness (compatibility) cost $E_s(l_q,l')$ between l' and l_q, 2) the self-judgment cost $E_d(l')$ of p about being assigned l', and 3) the opinion $M_{pq}^{t-1}(l')$ from neighbors expect q ($\mathbf{N}_p \backslash q$) about assigning l'.

All nodes exchange messages (opinions) about the label assignment and, through iterations, nodes far away from p can influence p's label assignment. According to [3], we define a single iteration as that of propagating a message from the top-left node to the bottom-right one, and then propagating in the opposite direction. After an enough number of iterations, say K, the label of p is determined based on the local likelihood and the messages from the neighbors as shown in Fig. 2(b):

$$l_p = \arg\min_{l \in L} \left\{ E_d(l) + \sum_{p' \in \mathbf{N}_p} M_{p'p}^K(l) \right\}. \quad (3)$$

The labels thus obtained correspond to a strong local minimum of the energy function (1). BP has several advantages for hardware implementation. Firstly, it is highly parallel because each node can operate independently in message passing. Secondly, it only uses simple operations such as additions and comparisons. Third, the memory access is regular. If we update the message sequentially, the required input data can be streamed into the processor with ease.

However, there is an issue related to the extensive computation required by BP. In each iteration, a image of N pixels with 4-connected neighborhood takes $4N$ message computations. According to Eq. (2), a message needs $L(3+L)$ additions and $L(L-1)$ comparisons. The constant factor 3 is included to account for the reuse of some partial summation results, as described in [7]. The computational complexity is proportional to $\mathcal{O}(NL^2)$. In stereo estimation of a VGA-sized image pair with 60 disparity values, a single BPM iteration requires 4.64G additions and 4.35G comparisons.

Therefore, fast message computation strategies must be applied. In [7], a two-pass algorithm with $\mathcal{O}(NL)$ complexity is proposed. This method only needs $(3+2)L$ additions and $2(L-1)$ comparisons to construct a message. However, it induces sequential dependency, which makes the parallel processing impossible. In hardware implementation, the cycle count and the latency linearly increase with L.

In order to allow for large discontinuities in the labeling, the cost function stops growing after the difference becomes large. A popular smoothness term in stereo algorithms is the generalized Potts model. It is a truncated linear function with the cost linearly increases with the distance between the labels p and q up to some level:

$$E_s(l,l') = \min_{l' \in L} (\lambda |l_q - l'|, \lambda T), \quad (4)$$

where λ is the weight of the smoothness cost, and T controls when the cost should stop increasing (Fig. 3).

3. FAST MESSAGE COMPUTATION ALGORITHM

We attack the computational issue at the algorithm level by proposing an efficient message computation method. The proposed hardware-oriented algorithm produces the results identical to the original method. We will then present the corresponding parallel PE design for the algorithm.

The pseudo codes of different message construction methods are listed in Table I, II, and III. All functions load three incoming messages Mu, Mr, Ms, the local data term D, and the smoothness term V (tabularized $E_s(l, l')$), and generates the outgoing message Mq. The loops that can be performed in parallel are indicated. Let D_A and D_C denote the latency of the adder and comparer respectively. The total

Table I. Message computation flow in original BP

BuildMessageOriginalParallel(Mq, Mr, Ms, Mu, D, V)	*Latency*
for l = 0,…, L−1 **in parallel** H[l] = (Mu[l]+Mr[l])+(Ms[l]+D[l]); **End**	$2D_A$
for l = 0,…, L−1 **in parallel** **for** m = 0,…, L−1 **in parallel 1 ADD** M_part[l][m] = H[m]+ λV[m][l]; **end** **end**	D_A + $log_2(L)D_C$
for l = 0,..., L−1 **in parallel** **for** m = 0,…,L−1 **in parallel $\log_2$(L) CMP** Mq[l] = min(M_part[l][m], Mq[l]) **end** **end**	

Table II. Fast message computation in [7]

BuildMessageEfficient(Mq, Mr, Ms, Mu, D, V)	*Latency*
for l = 0,…, L−1 **in parallel** H[l] = (Mu[l]+Mr[l])+(Ms[l]+D[l]); **end**	$2D_A$
for l = 1,…, L−1 **L-1 ADD and L−1 CMP** H[l] = min(H[l-1]+λ, H[l]); **end** **for** l = L−2,…, 0 **L−1 ADD and L−1 CMP** H[l] = min(H[l+1]+λ, H[l]); **end**	2(*L*-1)× (D_A+D_C)
//**Minimum truncation phase** M_min=M[0]; **for** l = 1,…, L−1 **in parallel $\log_2$(L) CMP** M_min = min(H[l], M_min); **end** **for** l = 0,…, L−1 **in parallel 1 ADD and 1 CMP** Mq[l] = min(H[l], M_min + λT); **end**	$log_2(L)D_C$ + D_A+D_c

latency for each stage in the pseudo code is shown at right. Note that in the pseudo codes, we include all possible parallelism to not only our algorithm, but also the original BP and the efficient BP algorithm in [7].

In original BP, the messages can be calculated by generating all L^2 hypotheses first, and then finding the L minimal final entities from them in parallel. The latency of this operation is $\log_2(L)D_C$. However, it requires L^2 temporal registers and operators. If we give up the parallelism, it takes $\mathcal{O}(L^2)$ cycles.

The software-efficient algorithm in [7] is shown in Table II. The entities are updated sequentially in two forward-backward passes and then truncated by the threshold. The sequential operation is suitable for software but causes a huge latency and low throughput in hardware. In hardware implementation, it is actually much slower than the parallelized original BP.

3.1. Proposed Algorithm

Table III. Proposed fast, hardware-oriented message computation algorithm

BuildMessageProposed(Mq, Mr, Ms, Mu, D, V)	*Latency*
for l = 0,…, L−1 **in parallel** H[l] = (Mu[l]+Mr[l]) + (Ms[l]+D[l]); **end**	$2D_A$
//**The local minimum cost comparison phase** **for** l = 0,…, L−1 **in parallel 1 ADD** **for m = −T+1,⋯, T−1 in parallel** M_part[l][l + m] = H[l + m]+ λV[l + m][l]; **end** //for l **for** l = 0,…, L−1 **in parallel** **for m = −T+1,⋯, T−1 in parallel $\log_2$(2T−1) CMP** M[l] = min(M_part[l][l + m], M[l]); **end** **end**	D_A+ $log_2(2T\text{-}1)D_C$
//**The global minimum threshold phase** **for** l = 0,…, L−1 **in parallel $\log_2$(L) CMP** M_min = min(H[l]+λT, M_min); **end**	$log_2(L)D_C$+D_A
//**Global truncation phase** **for** l = 0,…, L−1 **in parallel 1 ADD and CMP** Mq[l] = min(M[l], M_min); **end**	D_A+ D_C

We find that in the stereo estimation, the truncated linear function is commonly applied for the smoothness term. This function increases as the distance between the labels l and l' increases up to a certain level, and then becomes fixed to a threshold value T, which is typically 1 or 2.

Because all truncated values are constant, the L-to-1 comparisons for the i-th entity can be reduced to $(2T-1)$-to-1 comparisons from $i-T+1$ to $i+T-1$ neighboring labels, plus one comparison with the global minimum threshold, as shown in Table III. All redundant additions and comparisons can be removed. Take Fig. 3 as the example, all non-global minimum thresholds (green lines) are larger than the global minimum one (blue line) and can be skipped in the comparison process.

Furthermore, the local minimum cost comparison and the global minimum threshold finding can be done in parallel. All the above results in an efficient hardware design with reduced data access and fan-out. This method can also be generalized to handle many robust functions commonly used as the smoothness term, as described in [4].

3.2. Processing Element Design

The data flow and the processing element designs are present in Fig. 4 and Fig. 5, respectively. In the proposed method, only neighboring $2T-1$ hypotheses are generated and compared. In typical cases, T=2, and thus only three hypotheses are compared. Then the minimum hypothesis is then truncated by the global minimum threshold, which is generated once and served for all entities. The global mini-

mum threshold finding and local minimum cost comparison can be performed in parallel.

4. COMPARISON AND DISCUSSION

The complexity-latency-throughput comparisons of three methods are listed on Table IV. Compared with the original message computation, the proposed algorithm has a much lower complexity but still preserves the high throughput and the low latency. Only an additional D_c latency is included. On the contrary, the efficient BP method [7] has the same complexity, but has a much larger latency and low throughput. If we measure the operation count, the proposed algorithm and the efficient BP method have similar performance. Note that the proposed acceleration is a low-level technique is independent of the high-level applications.

We synthesize the circuits by using the UMC90nm library with the critical path constraint 10ns. Compared with the original message computation method, the proposed method reduces 94.4% gate counts, as shown in Table V. Moreover, the proposed algorithm has much less fan-outs than the original method does, and hence the latency of the critical path is greatly reduced.

Finally, we apply our algorithm to two different massage passing schemes, the BPM [3] and the tile-based BP proposed in [5]. The software execution time is shown in Table VI. We can see that the execution time is accelerated by a factor of 5.53 in BPM and 3.93 in tile-based BP. This means our algorithm is also beneficial for the software implementation.

5. CONCLUSION

Belief propagation is a popular global energy minimization technique. It can obtain a much better solution than the local optimization algorithm, but is also more computational intensive. In this paper, we have proposed a fast message computation algorithm for BP. By only generating and comparing a small number of hypotheses and the global minimum threshold, the proposed algorithm can greatly reduce the computational complexity. The low latency also enables the high throughput in parallel processing. We hope this new design can make the belief propagation a more practical algorithm for real-time applications.

6. REFERENCES

[1] Y. Boykov, O. Veksler, and R. Zabih, "Fast Approximate Energy Minimization via Graph Cuts," in *IEEE Trans PAMI*, vol. 23, no. 11, pp. 1222-1239, 2001.

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

[3] R. Szeliski, et al., "A Comparative Study of Energy Minimization Methods for Markov Random Fields with Smoothness-Based Priors," *IEEE Trans PAMI*, vol. 30, no. 6, pp. 1068-1080, 2008.

[4] C.-C. Liang, C.-C. Cheng, C.-K. Liang, Y.-C. Lai, L.-G. Chen, and H. H. Chen, "Hardware-Efficient Belief Propagation," *submitted*.

[5] C.-C. Cheng, C.-K. Liang, Y.-C. Lai, H. H. Chen, L.-G. Chen, "Analysis of Belief Propagation for Hardware Realization," in *Proc. SiPS*, 2008.

[6] J. Sun, N. N. Zheng, and H. Y. Shum, "Stereo Matching Using Belief Propagation," *IEEE Trans. PAMI*, vol. 25, no. 7, pp. 787-800, July 2003.

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

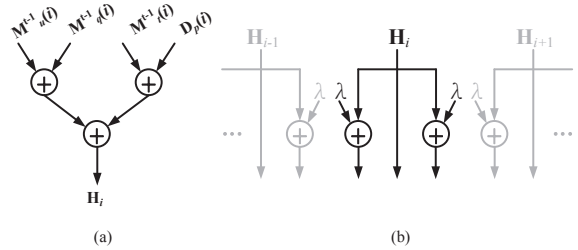

Fig. 4. The data paths for (a) the generation of H[i] and (b) the generation of hypotheses.

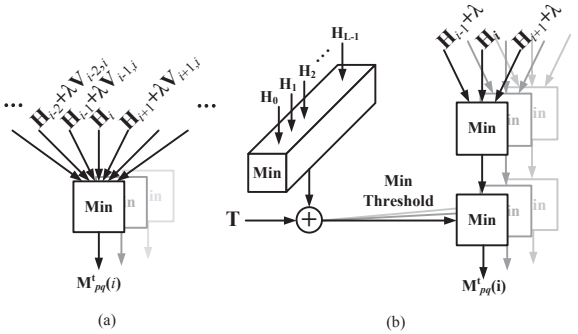

Fig. 5. (a) The PE and data flow of the traditional message construction. (b) The proposed PE and data flow. The grey items have the same structure but different inputs and work in parallel.

Table IV. Overall comparison of different methods

	Original [3]	Efficient BP [7]	Proposed
Complexity	$\mathcal{O}(L^2)$	$\mathcal{O}(L)$	$\mathcal{O}(L)$
Latency under max parallelism	$3D_A+$ $\log2(L)D_C$	$3D_A+$ $2(L-1)*(D_A+D_C)+$ $2\log_2(L)D_C+D_C$	$3D_A+$ $\log2(L)D_C+$ D_C
Throughput	L	1	L
Operations for L=64, T=2	4480	576	572

Table V. Comparison of gate count

L=64 UMC90nm@10ns	Original [3]	Proposed	Reduction ratio
Gate count	533.8k	30k	94.38%

Table VI. Comparison of the software execution time

CPU: Core2Duo E8400 Image Size: 450x375 ,L=64	BPM [3]	Tile-based BP [4]
Original algorithm	20.984s	14.688s
Proposed algorithm	3.797s	3.733s
Speedup ratio (original/fast)	5.53	3.93