Hardware-Efficient True Motion Estimator Based on Markov Random Field Motion Vector Correction

Fu-Chen Chen[1], Yung-Lin Huang[1], and Shao-Yi Chien[2]
Media IC and System Lab
[1] Graduate Institute of Networking and Multimedia
[2] Graduate Institute of Electronics Engineering and Department of Electrical Engineering
National Taiwan University
MD-726, No. 1, Sec. 4, Roosevelt Rd., Taipei 106, Taiwan, R.O.C.

***Abstract*—True motion estimation is a well-known technique to find the true object motion trajectory in a video, and it has a lot of applications in computer vision and display systems. However, if the target frame size becomes large, many new design challenges are introduced, such as huge computation, large bandwidth and large on-chip SRAM size requirements. Within the consideration of both algorithm and architecture, we develop a true motion estimator with ±128x±128 search range for video systems with Full-HD (1920x1080) resolution. The PSNR evaluation shows that our algorithm is better than other three existing algorithms. For hardware implementation, we use Verilog-HDL and synthesize it by SYNOPSIS Design Compiler with UMC 90nm cell library. The implementation works at 300MHz frequency, and it shows that there are total 76% bandwidth reduction, 66% cycle reduction and 88% on-chip SRAM reduction with the proposed ping-pong two-way scheduling and motion vector grouping techniques.**

INTRODUCTION

Unlike conventional motion estimation designed for video coding, the purpose of true motion estimation is to find the real motion presenting objects' movement [1], not just to reduce the residual energy of each block. It has a lot of applications in computer vision and display systems, and is often a time and area consuming part of the system since the number of required candidates of block matching is usually large. Recently, many true motion estimation algorithms are proposed with decreasing the number of candidates [1][2][3]. These algorithms often utilize the characteristics of spatial and temporal coherence in motion vector field.

To make motion vector field more reliable for presenting objects' movement, some motion vector post-processing techniques are employed after motion estimation. Since the true motion vector field has the characteristics of spatial and temporal coherence, some simple operations such as median filter or weighted averaging can provide acceptable results. To achieve better quality, the motion vector field can be modeled as a 3D Markov Random Field (MRF) and formed as an energy minimization problem [4]. Markov Random Field is a theoretical modeling method based on Bayesian's framework, applied to computer vision algorithms for many years such as optical flow or true motion estimation [4]. The global energy minimization is a NP-complete problem so many fast algorithms for finding local minimum are proposed [5].

While the frame sizes of video systems become large, many new design challenges are introduced, such as huge computation, large bandwidth and large on-chip SRAM size requirements. In this paper, a hardware-efficient true motion estimator is proposed based on Markov Random Field motion vector correction. The motion estimator is optimized in both algorithm and hardware architecture levels to achieve the high specification of Full-HD (1920x1080) 24fps with ±128x±128 in search range. The proposed key techniques are shown in the following sections.

PROPOSED TRUE MOTION ESTIMATION ALGORITHM BASED ON MARKOV RANDOM FIELD

First of all, a low-complexity true motion estimation algorithm is proposed. The block size is set as 32x32, and the matching criterion is 8x8 multilevel successive elimination algorithm (MSEA) [6] for hardware complexity consideration. Moreover, the search range is set as ±128x±128 for Full-HD (1920x1080) videos. Furthermore, we perform motion vector correction based on Markov Random Field modeling with a low-cost but robust version of iterated conditional mode (ICM) minimization.

Predictive Square Search Motion Estimation

This motion estimation algorithm is very similar to a hybrid search algorithm with four step search (4SS) [7] and three step search (3SS) [8]. The search pattern is demonstrated in Figure 1(a). At first, a predictor is given by the median of three neighboring (left, up and upper-right) motion vectors. Then a 4-step square search pattern centering on the predictor is employed. If the minimum distortion appears at the center or its value is smaller than the threshold, the predictor will be regarded as good and proceed to apply 2-step and 1-step square patterns for converge, like 3SS. Otherwise, we go back to the origin and search motion vector like 4SS but with an 8-step square pattern. If the minimum distortion found at the center of a 8-step square pattern, 4-step, 2-step and 1-step square patterns are employed for converge.

The pseudo code of the proposed search algorithm is shown as follows. Here *SP* means square pattern, and ε means minimum distortion of the applied *SP*. *MV* means motion vector value, which is used as the center of *SP,* and the *threshold* is equal to 1024 in our implementation.

Pseudo Code of the Proposed Search Algorithm	
Step. 1	Set *MV* = median of three neighboring blocks' *MVs*
Step. 2	Apply *4-step SP* on *MV*
	if ε is found at the center or ε < *threshold*
	Apply *2-step* and *1-step SPs* for converge
	Else
	Set *MV* = origin, go to Step. 3
Step. 3	Apply *8-step SP* on *MV*
	if ε is not found at center
	Set *MV* = the position with ε, repeat Step. 3
	Else
	Apply *4-step*, *2-step* and *1-step SPs* for converge

978-1-4577-2081-9/12/$26.00 ©2012 IEEE

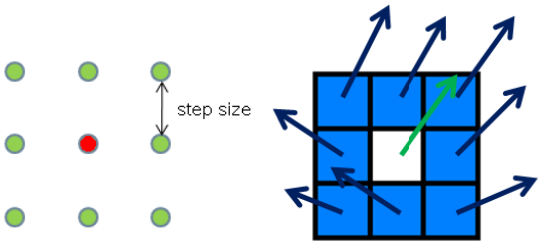

FIGURE 1. PATTERN USED IN THE PROPOSED TRUE MOTION ESTIMATION ALGORITHM. (A) SQUARE PATTERN. (B) NINE CANDIDATES OF NEW MOTION VECTOR.

8x8 MSEA is chosen as the matching criterion for block-matching. The 8x8 MSEA can be regarded as the down-sampled version of sum of absolute difference (SAD). As shown in Figure 2, each 32x32 block is divided into 16 8x8 sub-blocks, and each sub-block is sumed up. Next, the 16 absolute differences of the summed up sub-block pairs are accumulated to derive the 8x8 MSEA.

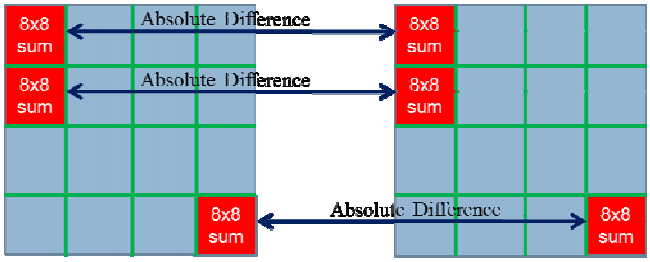

FIGURE 2. ILLUSTRATION OF 8X8 MSEA.

The original purpose of calculating MSEA is for fast full search. However, we found that employing MSEA with square patterns reduces lots of computation and bandwidth cost in hardware design, especially when the step size matches the sub-block size. Moreover, the generated motion vector field is almost the same as that generated with SAD as the matching criterion.

Markov Random Field Motion Vector Correction

After motion estimation, the rough motion vector field is formed, but some motion vector outliers exist. In this step, we process the blocks in raster order to refine the motion vectors based on MRF modeling. Among the energy minimization methods, we choose the simplest ICM with selected candidates.

Unlike the general ICM that selects all candidates in a search range, we only choose nine candidates adjacent to the block. In the literature, it shows that there is a very high probability for a block to find its true motion from nearby block's motion vectors [9]; therefore, choosing neighboring nine candidates is enough. In addition, another benefit of choosing neighboring nine candidates is preventing over-smoothing, and the complexity is also lower than selecting all candidates in the search range. The details are shown as follows.

For a block, eight neighboring motion vectors and itself motion vector are chosen as nine new motion vector candidates, as shown in Figure 1(b). Afterwards, the corresponding MRF energy for each candidate is computed as follows,

$$MRF\ energy_{candi.} = 8x8\ MSEA + weight \times \sum_{all\ neigh.} \left| MV_{candi.} - MV_{neigh.} \right|$$

Finally, the one with the smallest MRF energy is selected from these nine candidates as the new motion vector of this block as follows,

$$new\ MV = \underset{MV_{candi.}}{arg\,min.}\ MRF\ energy_{candi.}$$

In our design, the weight is set as 48. Moreover, running ICM for three iterations is enough to remove most motion vector outliers.

PROPOSED HARDWARE-EFFICIENT ARCHITECTURE OF TRUE MOTION ESTIMATOR

There are three design challenges to design the hardware architecture of the proposed true motion estimation algorithm for Full-HD videos. The first challenge encountered is the requirement of large on-chip SRAM. To support ±128x±128 search range, the on-chip SRAM size will be large. Even if scheme C data reuse is adopted [10], the SRAM size is still $(128 + 128 + 32)^2 = 82944$ Bytes. Second, to fetch data for the next block, the bandwidth consumption is (128 + 128 + 32) x 32 x 2040 (total number of blocks) = 18.8MB for one frame. Third, the cycles for fetching are (128 + 128 + 32) x 32 / 16 = 576 cycles per block. All of the resource requirements above are too much to be feasible.

On-chip SRAM Reduction

By employing the characteristics of the proposed motion estimation algorithm, the resource requirement can be greatly reduced. For 4, 2 and 1-step convergence in the algorithm, we prepare a set of SRAM called "M1" with range ±8x±8 to save all the possible required pixels. Note that although the range of ±7x±7 is already enough for motion estimation, we set the size to be ±8x±8 to share with other modules.

For 8-step re-estimation from origin with 8x8 MSEA criterions, scheme C data reuse can be employed on the 8x8 sub-block summation values (SUM) with much less resource consumption, which is stored in a set of SRAM called "O1." With this scheme, the bandwidth consumption is reduced to around 7.2MB for one frame for a worst-case video sequence. The required size of M1 and O1 memory is shown in Figure 3.

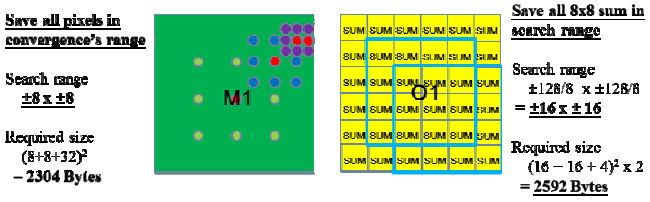

FIGURE 3. SRAM USAGE FOR MOTION ESTIMATION.

Ping-pong Two-way Scheduling

Obviously, there are many data and operation dependencies between in the proposed motion estimation. That is, each block cannot be processed until the motion vector of the previous block is determined. It will introduce pipeline bubbles to reduce the hardware utilization with direct scheduling, as shown in Figure 4.

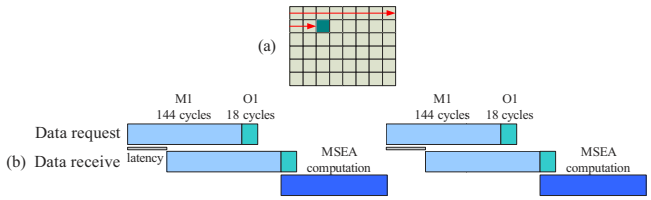

FIGURE 4. DIRECT SCHEDULING.
(A) RASTER SCAN ORDER. (B) PIPELINE BUBBLES.

To eliminate the dependencies, a ping-pong two-way scheduling is proposed, where an additional SRAM pair of "M2" and "O2" like original M1 and O1 is introduced for data interleaving. The ping-pong means that one of the pair is used for computing while the other is used for data pre-fetching. The two-way means one of the pair is running raster scan and the other is running inverse raster scan, as shown in Figure 5(a). By doing so, the pipeline bubbles are eliminated with ping-pong data pre-fetching, as shown in Figure 5(b). For the best balance between data fetching and MSEA computing, the cycles consumed for 4, 2 and 1-step convergence should be about 18 + 144 = 162 cycles.

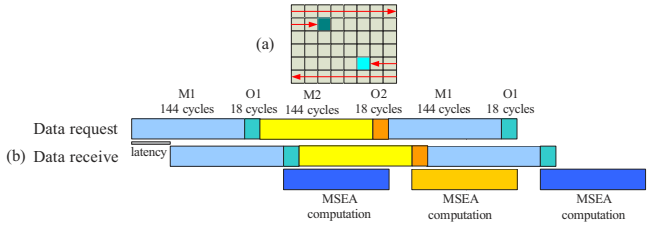

FIGURE 5. PING-PONG TWO-WAY SCHEDULING.
(A) TWO-WAY SCAN ORDER. (B) PING-PONG USAGE WITHOUT PIPELINE BUBBLES.

In our proposed motion estimation, we only read the pixels needed for 4, 2, 1-step convergence and 8x8 sum for the 8-step square pattern with a huge amount of bandwidth and SRAM size reduction. The final synthesized SRAM size is 48 address x 128 bits x 3 banks x 2 = 4,608 Bytes for M1 and M2, and 84 address x 16 bits x 16 banks x 2 = 5,376 Bytes for O1 and O2. The SRAM size is reduced from 82,944 Bytes to 4,608 + 5,376 = 9,984 Bytes, and the bandwidth is reduced from 18.8MB to 7.2MB for one frame. In addition, the proposed SRAM is also shared with the other modules to further increase the hardware utilization.

Motion Vector Grouping

After motion estimation, the 8x8 MSEA value of each block will be written out to DRAM for MRF energy computing. However, the 8x8 MSEA values of the other eight candidates are still unknown and need to be calculated, and the motion vectors of these 8 candidates may not be the same. If we fetch pixels candidate-by-candidate for these 8x8 MSEA computing, it will consume 16MB per iteration and 512 cycles per block. This direction implementation will deplete computing and memory bandwidth and is not feasible.

By employing the characteristics of motion vector field computed from our motion estimation algorithm, neighboring motion vectors are similar and many required pixels are overlapped. As shown in Figure 6, if we can determine the center motion vector of a group, and then fetch all the pixels around this center motion vector into M1 (M2), all the motion vectors with distances to the center motion vector smaller or equal to 8 (search range of M1 and M2) can be calculated. Therefore, a motion vector grouping technique is proposed for bandwidth reduction. The group size must be bigger than two for bandwidth gain, and there are at most two groups in our design.

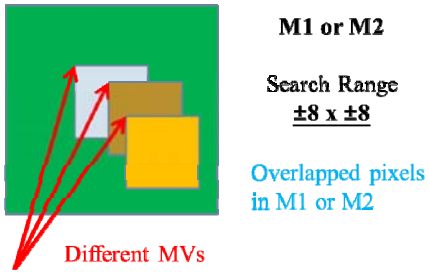

FIGURE 6. OVERLAPPED PIXELS OF DIFFERENT MOTION VECTORS IN M1 OR M2.

The eight candidates are regarded as eight nodes with twenty-eight edges for all possible connections between them, and the corresponding motion vector distance is calculated for each edge. It actually consumes no computation overhead since they are part of MRF energy. The edges with distance smaller or equal to eight are labeled, and we count the total number of labeled edges connecting to the nodes. The node with maximum number is the center motion vector of this group, and the nodes with labeled edges connecting to the center motion vector are the members of this group. After generating group 1, the nodes and edges of this group are removed, and group 2 can be then generated with the same process. The nodes that are not grouped called non-group.

Architecture of Grouping

Figure 7 shows the proposed hardware architecture for motion vector grouping. We compute the motion vector distances one-by-one, and accumulate them into "Total MV dis. Registers" for MRF energy computing. Meanwhile, it is judged whether the distances are smaller or equal to 8, and the corresponding edge registers are labeled. After that, we perform the proposed grouping algorithm and push the center and member motion vectors into three types of queues.

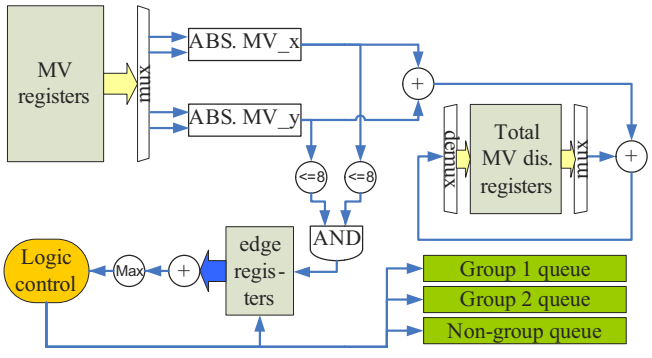

FIGURE 7. PROPOSED ARCHITECTURE OF GROUPING

The proposed motion vector grouping algorithm and its architecture are also used for distance energy computation and median filter. The cycle is reduced from 512 x 3 = 1,536 to 320 + 82 + 57 = 459 cycles per block for 3 MRF iterations in the worst case. Also, the bandwidth is reduced from 16MB x 3 = 48MB to 7.2 + 1.1 + 0.3 = 8.6MB for 3 MRF iterations.

PERFORMANCE EVALUATION

Several Full-HD (1920x1080) video sequences are used for performance evaluation. To evaluate the performance of the proposed true motion estimator, we employ the derived motion vector for motion compensated frame interpolation (MCFI) to show the quality indirectly: with the original odd frames, we compute the PSNR values between the interpolated even frames and the original even frames which are treated as the ground truth. Three state-of-the-art algorithms are selected for comparison. *Yang* [11] derives motion vectors from H.264, performing OBME and OBMC; *Percept.* [12] also extracts motion vectors from H.264, ignoring motion vectors that are perceptually unapparent; *GME* [13] performs global motion estimation and sub-block division. One of the experimental results for sequence "Transformer" is shown in Figure 8. As the result shows, our proposed algorithm outperforms other three algorithms. Similar results can also be found for other video sequences.

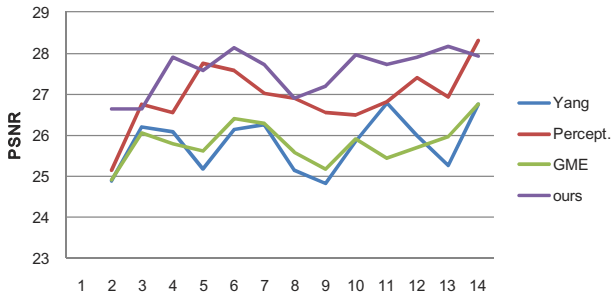

FIGURE 8. PSNR EVALUATION OF MCFI. (SEQUENCE: TRANSFORMER)

IMPLEMENTATION

We use Verilog-HDL for hardware implementation, and synthesize it by SYNOPSIS Design Compiler with UMC 90nm cell library. It works at 300MHz in frequency with a 128-bit bus. In brief, it provides true motion estimation based on Markov Random Field motion vector correction with ±128x±128 search range and supports video systems with Full-HD (1920x1080) resolution. Table 1 shows the overall resource saving and characteristics of the proposed architecture. Lots of cycles, bandwidth and on-chip SRAM are saved by our proposed hardware design techniques.

TABLE 1. OVERALL RESOURCE SAVING OF THE PROPOSED ARCHITECTURE.

	ME		MRFx3		Total
	Direct	Proposed	Direct	Proposed	Reduction
Cycles (per block)	576	266	1536	459	66%
	All of the schedules are tight				
Bandwidth	18.8MB	7.2MB	48MB	8.6MB	76%
	Pixel arrangement				
SRAM	82,944 Bytes	9,984 Bytes	Shared with ME		88%

CONCLUSION

In this paper, we first propose a predictive square search algorithm for true motion estimation with 32x32 block size, 8x8 MSEA criterions and ±128x±128 search range. Moreover, MRF model is employed to perform motion vector correction and minimize the energy using low-cost ICM. With the careful arrangement of SRAM, proposed ping-pong two-way scheduling, motion vector grouping and its architecture, we can reduce totally 76% bandwidth, 66% cycles and 88% on-chip SRAM size. Also, the PSNR evaluation and implementation results indicate that the proposed algorithm and architecture consumes reasonable amount of resources but still maintains well performance.

REFERENCES

[1] G. de Haan, P.W.A.C. Biezen, H. Huijgen, and O.A. Ojo, "True-motion estimation with 3-D recursive search block matching," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 3, no. 5, pp. 368 - 379, 388, Oct. 1993

[2] J. Wang, D. Wang, and W. Zhang, "Temporal compensated motion estimation with simple block-based prediction," *IEEE Trans. Broadcasting*, vol. 49, no. 3, pp. 241-248, Sept. 2003

[3] A. M. Tourapis, "Enhanced predictive zonal search for single and multiple frame motion estimation," in *Proc. Visual Communications and Image Processing*, page 1069-1079, Feb. 2002

[4] D. Wang, L. Zhang, and A. Vincent, "Motion-Compensated Frame Rate Up-Conversion—Part I: Fast Multi-Frame Motion Estimation," *IEEE Trans. Broadcasting*, vol. 56, no. 2, pp. 133-141, June 2010

[5] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother, "A Comparative Study of Energy Minimization Methods for Markov Random Fields," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 30, no. 6, pp. 1068 - 1080, June 2008

[6] X.Q. Gao, C.J. Duanmu, and C.R. Zou, "A multilevel successive elimination algorithm for block matching motion estimation," *IEEE Trans. Image Processing*, page vol. 9, no. 3, pp. 501 - 504, Mar. 2000.

[7] L.-M. Po and W.-C. Ma, "A novel four-step search algorithm for fast block motion estimation," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 6, no. 3, pp. 313 - 317, Jun. 1996.

[8] R. Li, B. Zeng, and M. L. Liou, "A new three-step search algorithm for block motion," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 4, No. 4, pp. 438-442, Aug. 1994.

[9] Y.-L. Huang, Y.-N. Liu, and S.-Y. Chien, "MRF-based true motion estimation using H.264 decoding information," in *Proc. IEEE Workshop on Signal Processing Systems*, Oct. 2010.

[10] J.-C. Tuan, T.-S. Chang, and C.-W. Jen, "On the data reuse and memory bandwidth analysis for full-search block-matching VLSI architecture," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 12, no. 1, pp. 61-72, Jan. 2002.

[11] Y.-T. Yang, Y.-S. Tung, J.-L. Wu, "Quality enhancement of frame rate up-converted video by adaptive frame skip and reliable motion extraction," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 17, no. 12, pp.1700 – 1713, Dec. 2007

[12] Y.-N. Liu, Y.-T. Wang, S.-Y. Chien, "Motion Blur Reduction of Liquid Crystal Displays Using Perception-Aware Motion Compensated Frame Rate Up-Conversion," in *Proc. IEEE Workshop on Signal Processing Systems*, Oct. 2011

[13] K.-Y. Hsu, S.-Y. Chien, "Frame rate up-conversion with global-to-local iterative motion compensated interpolation," in Proc. IEEE International Conference on Multimedia and Expo, June 2008