A FLEXIBLE HARDWARE-ORIENTED FAST ALGORITHM FOR MOTION ESTIMATION*

Fengqi Yu and Alan N. Willson, Jr.

Integrated Circuits and Systems Laboratory
University of California, Los Angeles
Los Angeles, CA 90095

ABSTRACT

This paper discusses the design of a fast algorithm for motion estimation with emphasis on hardware cost considerations, real-time application, network adaptation, and flexibility. To achieve the best trade-off among hardware cost, computational complexity, and distortion performance, we propose a multi-stage pixel-subsampling motion estimation algorithm. The algorithm has a lower hardware cost than Liu's subsampling algorithm and the three-step hierarchical search algorithm (3SHS) in terms of data flow control, I/O bandwidth, and regularity. Its computational complexity is close to that of 3SHS and its distortion performance, which is better than that of Liu's algorithm and 3SHS, is close to that of full search.

1. INTRODUCTION

To evaluate a fast motion estimation algorithm for hardware implementation, two factors should be considered: performance and cost. Speed (low computational complexity) and distortion are two important performance measures that have been addressed in most papers, however hardware implementation cost has not received nearly as much attention by algorithm designers. Most current fast algorithms [1-5] actually reduce computational complexity by increasing hardware cost. More attention must be paid to ascertaining the best trade-off between computational complexity and hardware cost. A second problem of some current fast algorithms is that they can only give an improved average speed, not a better worst-case performance. Therefore these algorithms are inappropriate for most real-time applications. A third problem is that source coding is isolated from network traffic conditions. One last problem is that some successful fast algorithms for high-frame-rate applications fail for very-low-frame-rate situations due to deteriorating performance and/or higher complexity. The motivation of this paper is to present a flexible hardware-driven fast motion estimation algorithm for various frame-rate applications, having the capability to adapt its performance in keeping with network traffic constraints. The paper is organized as follows. In Section 2, the proposed algorithm is described. In Section 3, the comparison of computational complexity and hardware complexity of various algorithms is reported. In Section 4, the comparison of distortion performance is presented. Finally, Section 5 draws the conclusion.

[1]This work was supported by the National Science Foundation under Grant MIP-9632698 and by the Office of Naval Research under Grant N00014-95-1-0231.

2. THE PROPOSED ALGORITHM

To provide a compromise between hardware cost, computational complexity, and distortion performance, we introduce a multi-stage subsampling motion estimation algorithm, where the decimations of the search window and the current block are optimized separately. As an example, we discuss our algorithm for low-frame-rate applications in which the search range is [-16, 16).

By trying various decimation factors, we have found that the optimal solution for low-frame-rate applications seems to be a three-stage search. In the first stage, the decimation factor is 4:1 for the current block and 16:1 for the search window. In the second stage, the decimation factor is 4:1 for both the current block and the search window, with the search range [-8, 8). The last stage is a full search with all pixels in the current block involved, in which the search range is $[-p_f, p_f)$, where p_f is adapted by the network traffic condition and performance requirements. For busy traffic and/or high performance p_f is chosen to be large. The larger the search window, the better the motion vector—which results in fewer bits being sent to the network. (When the network is busy, the coding layer can take more time to do a better job, thereby sending fewer bits out on the network.)

From a hardware cost point-of-view, we distribute the input bandwidth almost evenly to each stage; there-

fore the overall bandwidth is kept small. The data flow control complexity is small because of the algorithm's regularity. From a computational complexity point-of-view, the initial stages are made cheaper by a large decimation of pels in the search window, and final fine-adjustment stages are made cheaper by reducing the search range. From a performance point-of-view, the search positions are evenly distributed in a search window, which makes the mean-absolute-error search unlikely to be trapped in a local minimum.

For high-frame-rate applications with a search range of [-8, 8), using two stages seems optimal in terms of hardware cost and distortion performance. For an even larger search range, say [-32, 32), four stages may be chosen with properly selected decimation factors and search ranges for each stage.

3. COMPARISON OF COMPLEXITIES

For the hardware design of a motion estimator, two hardware costs should be considered. One is manufacturing cost, which includes chip area and chip packaging. The other is design cost. Chip area is determined by the number of processing elements, data flow control elements, local memory, and interconnections among modules and between a module and external memories. The interconnections depend highly on the memory access rate and the regularity of an architecture. In other words, the overall chip area is determined by chip core area and I/O bandwidth. Since the output bandwidth is much smaller than the input bandwidth, only the input bandwidth is considered, and it can be estimated by [6] $BW = N/(PE \times f_c)$; where BW is the input bandwidth in pel/cycle; N is the total number of input pixels for each current block; PE is the number of processing elements; and f_c is the number of cycles for obtaining the motion vector of a current block. As for design cost, it is determined by architectural complexity, algorithm regularity and data flow control.

To make an overall evaluation of an algorithm, we choose three criteria: computational complexity (CC), processing element (PE) and associated hardware requirements, and input bandwidth (BW). For ease of comparison, we chose a 16 × 16 2-D array architecture for all algorithms considered. In order to obtain high performance for Liu's algorithm [4], parallel processing of its three steps had to be adopted by adding extra hardware. The relative complexities of major motion estimation algorithms and those of the proposed algorithm are listed in Table I, where the search range is $[-p, p)$.

Computational complexity: The 3SHS and the proposed algorithm win for speed, especially for p=16. The computational complexity of the proposed algorithm is slightly higher than that of the 3SHS for p=16.

TABLE I. Comparison of complexities of various motion estimation algorithms

	$p = 8$			$p = 16$		
	CC	PE	BW	CC	PE	BW
FS	1.0	1.00	1.00	1.0	1.00	1.00
Liu's alg.	1/8.0	1.13	3.98	1/8.0	1.03	2.59
3SHS	1/10.2	1.00	6.26	1/31.0	1.00	10.32
proposed	1/10.2	1.00	3.98	1/25.0	1.00	4.30

Processing elements and associated hardware: Liu's algorithm needs an additional 13% in PEs to achieve an input bandwidth reduction and lower computational complexity for p=8.

Input bandwidth: The BW in Table I is the worst-case input bandwidth. Even though the input bandwidth of Liu's algorithm is the same as that of the proposed algorithm for p=8, the cost of achieving this performance is increasing data flow control complexity due to parallel processing of its second step and third step, and an additional 13% in PEs. It should be pointed out that Liu's algorithm is running at a lower speed, which makes its bandwidth look lower. For p=16, the input bandwidth of Liu's algorithm is slightly lower than that of the proposed algorithm, however its speed is only 1/3 that of the proposed algorithm.

Data flow control: The data flow control requirements of Liu's algorithm are much higher than those of the proposed algorithm due to the different pixel decimation patterns and the block decimation. The introduction of two parallel processings in Liu's algorithm makes its data flow control even harder to manage. The proposed algorithm is more regular than 3SHS, therefore its data flow control should be less complex than that of 3SHS.

4. COMPARISON OF DISTORTION PERFORMANCE

To fully test our algorithm, we used various kinds of test sequences in terms of image size, small object motion, fast object motion, and quick camera motion. The test sequences are the Claire sequence, salesman sequence, football sequence, and garden sequence. The size of the Claire and football sequences is 256 × 256 pixels/frame, the size of the salesman sequence is 360 × 288 pixels/frame, and the size of the garden sequence is 720 × 486 pixels/frame. All sequences were run for 100 frames with a frame rate of 30 frames/sec for the high-frame-rate test and 10 frames/sec for the low-frame-rate test. The mean-absolute-error criterion is used in our simulations. Table II lists the distortion performance comparisons for high frame rates with a search range of [-8, 8). The comparisons of the dis-

TABLE II. Comparison of distortion performance for various algorithms for high frame rates with p_f=1

	Full search		Liu's alg.		3SHS		Proposed	
	Entropy	PSNR	Entropy	PSNR	Entropy	PSNR	Entropy	PSNR
Claire	2.5867	39.9287	2.5888	39.8296	2.5933	39.8424	2.6019	39.8493
Salesman	2.9982	38.5220	3.0036	38.3634	3.0122	38.3873	3.0050	38.4651
Football	5.5066	22.5386	5.5579	21.9740	5.5746	22.0111	5.5420	22.3160
Garden	4.9840	26.8906	5.0064	26.6036	5.2092	25.7499	4.9779	26.8828

TABLE III. Comparison of distortion performance for various algorithms for low frame rates with p_f=1

	Full search		Liu's alg.		3SHS		proposed	
	Entropy	PSNR	Entropy	PSNR	Entropy	PSNR	Entropy	PSNR
Claire	3.0465	35.6249	3.0647	35.2697	3.0841	34.8512	3.0794	35.2613
Salesman	3.3313	34.7351	3.3494	34.1803	3.3720	34.1812	3.3522	34.5467
Football	6.0133	20.2526	6.1242	19.3200	6.2132	19.1334	6.1135	19.8949
Garden	5.8578	21.1784	5.9324	20.4175	6.2639	18.7226	5.8370	21.5957

TABLE IV. Performance of the proposed algorithm for various p_f values in terms of PSNR for high frame rates

	p_f=0	p_f=1	p_f=2	p_f=4
Claire	38.0079	39.8493	39.8982	39.9216
Salesman	37.6928	38.4651	38.5222	38.5370
Football	20.9426	22.3160	22.4270	22.5454
Garden	25.3057	26.8828	26.9964	27.0887

tortion performance for large motion vectors, with a search range of [-16, 16), are listed in Table III.

The distortion performance of the proposed algorithm is better than both Liu's algorithm and 3SHS. It is worth mentioning that the performance of the proposed algorithm shows a significant improvement for fast object motion sequences, quick camera motion sequences, and low-frame-rate applications. From Table III, we can see that the PSNR of the proposed algorithm can even be better than that of the full search. This happens because the search range of the last stage goes beyond the original search window for the garden sequence.

The performance of the proposed algorithm for various p_f values was also simulated and the results are presented in Table IV. We find a significant performance improvement from $p_f = 0$ to $p_f = 1$, and also from $p_f = 1$ to $p_f = 2$.

5. CONCLUSION

In this paper, a novel hardware-oriented fast block-matching algorithm for real-time applications has been proposed. The importance of the hardware cost of an algorithm is addressed. Considering hardware complexity and computational complexity, we have proposed a multi-stage subsampling fast algorithm for motion estimation. The proposed algorithm is very flexible. It can be adapted based on network traffic conditions and on distortion performance requirements. It can also be programmed for both high-frame-rate and low-frame-rate applications. The hardware costs of the proposed algorithm are lower than those of Liu's algorithm and 3SHS. Its computational complexity is close to that of 3SHS. Its distortion performance, which is better than that of Liu's algorithm and 3SHS, is close to that of full search.

References

[1] F. Dufaux and F. Moscheni, "Motion estimation techniques for digital TV: review and a new contribution," *Proc. IEEE*, vol. 83, pp. 858-876, June 1995.

[2] H.G. Musmann, P. Pirsch, and H. J. Grallert, "Advances in picture coding," *Proc. IEEE*, vol. 73, pp. 523-548, March 1985.

[3] M. Bierling, "Displacement estimation by hierarchical block matching," in *Proc. SPIE Visual Comm. and Image Proc.*, vol. 1001, pp. 942-951, 1988.

[4] B. Liu and A. Zaccarin, "New fast algorithms for the estimation of block motion vectors," *IEEE Trans. Circuits Syst. for Video Technology*, vol. 3, pp. 148-157, April 1993.

[5] M. Hoetter, "Object-oriented analysis-synthesis coding based on moving two-dimensional objects," *Signal Process.: Image Commun.*, vol. 2, pp. 409-428, Dec. 1990.

[6] M. C. Chen and A. N. Willson, Jr., "A parallel VLSI architecture with optimal input bandwidth utilization for the 3-step hierarchical search block-matching algorithm," in *Proc. 12th European Conference on Circuit Theory and Design*, pp. 239-242, August 1995.