Motion estimation based on an improved block matching technique

Tangfei Tao (陶唐飞)[1,2], Chongzhao Han (韩崇昭)[1], Yanqi Wu (吴艳琪)[1], and Xin Kang (康 欣)[1]

[1] *Institute of Synthetic Automation, School of Electronic and Information Engineering, Xi'an Jiaotong University, Xi'an 710049*

[2] *Key Laboratory of Education Ministry for Modern Design and Rotor-Bearing System, Xi'an Jiaotong University, Xi'an 710049*

Received August 16, 2005

An improved block-matching algorithm for fast motion estimation is proposed. The matching criterion is the sum of absolute difference. The basic idea is to obtain the best estimation of motion vectors by an optimization of the search process which can terminate the time-consuming computation of matching evaluation between the current block and the ineligible candidate block as early as possible and eliminate the search positions as many as possible in the search area. The performance of this algorithm is evaluated by theoretic analysis and compared with the full search algorithm (FSA). The simulation results demonstrate that the computation load of this algorithm is much less than that of FSA, and the motion vectors obtained by this algorithm are identical to those of FSA.

OCIS codes: 100.0100, 100.2000, 330.4150.

The block matching algorithm (BMA) has been widely used in motion estimation[1]. The basic idea of this technique is to partition the current frame image into blocks firstly, and then to find the best-matched block for each current block in a search area in the previous image frame according to the selected matching criterion. The optimal result can be obtained by a full search algorithm (FSA), however, the computation burden of FSA is too high, which limits its practical applications.

In order to resolve the heavy computation load of FSA, many fast searching algorithms have been developed. Some fast algorithms tend to restrict the searching in a subset, instead of all possible blocks by applying a special search strategy, such as two dimensional logarithmic search (TDLS)[2], three-step search (TSS)[3], diamond search (DS)[4], four–step search (FSS)[5], etc.. But these method cannot guarantee to get the global optimum due to the assumption that the matching error decreases monotonically as the searched point moves closer to global optimum. Some fast algorithms[6,7] try to subsample the current block and only partial pixels of the block are used in matching process. Since only a fraction of the pixels is used in the matching process, these methods cannot guarantee to get the best matching too. Some other fast algorithms[8,9] try to subsample the whole image and only partial instead of all blocks are taken into account, the motion vectors of these blocks can be gained directly and the motion vectors of the remaining blocks are gained by interpolating from its neighbor blocks. However, in fact, the motion fields are scene dependent, so the performance loss/gain is also scene dependent and unpredictable.

This paper presents an improved BMA for fast motion estimation, which can terminate the computation of matching evaluation as early as possible and eliminate search positions in the search window as many as possible. The proposed algorithm can achieve the same estimation accuracy as that of FSA, with much less computation burden than that of FSA.

The objective of the BMA is to find the best matched reference block (RB) in the previous frame within a search window for the current block (CB) in the current frame according to the selected matching criteria. Here, we try to find a method to constrain the search process in the search area while preserving the optimal solution for the motion vector. We assume that $N \times N$ is the block size, (u, v) is the candidate motion vector, p is the maximum allowed displacement, that is, the size of the search window is $(2p+1) \times (2p+1)$, $\mathrm{CB}(i, j)$ and $\mathrm{RB}(i, j)$ represent the intensities of the pixel with coordinate (i, j) in the current block and reference block, respectively. The sum of absolute difference (SAD) is used to measure the matching error between these two blocks.

$$\mathrm{SAD}(u, v) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} |\mathrm{CB}(i, j) - \mathrm{RB}(i+u, j+v)|,$$

$$\text{where } -p \le u, v \le p. \tag{1}$$

We denote $\mathrm{SAD}_j(u, v)$ to represent the SAD of jth row between the CB and RB with motion vector (u, v), which can be expressed as

$$\mathrm{SAD}_j(u, v) = \sum_{i=0}^{N-1} |\mathrm{CB}(i, j) - \mathrm{RB}(i+u, j+v)|. \tag{2}$$

So Eq. (1) can be rewritten as

$$\mathrm{SAD}(u, v) = \sum_{j=0}^{N-1} \mathrm{SAD}_j(u, v). \tag{3}$$

And we use $\mathrm{SAD}^k(u, v)$ to represent the SAD of the first k row(s) between the CB and RB with motion vector (u, v), which can be defined as

$$\mathrm{SAD}^k(u, v) = \sum_{j=0}^{k-1} \mathrm{SAD}_j(u, v). \tag{4}$$

Assume that $\mathrm{SAD}(m,n)$ is so far the minimum matching error obtained for the so far best matched block with motion vector (m,n), obviously, to search for a better matched candidate block, we are interested in those blocks with motion vector (x,y) for

$$\mathrm{SAD}^{N-1}(x,y) \le \mathrm{SAD}(m,n). \tag{5}$$

For any other candidate blocks, if k is smaller than $N-1$ and the $\mathrm{SAD}^k(x,y)$ between this block and the current block has exceeded the so far minimum matching error $\mathrm{SAD}(m,n)$ at that time, this candidate block with motion vector (x,y) is not eligible and should be rejected, and the remaining computation of matching error between the rest $N-k-1$ row(s) of these two blocks is unnecessary, i.e., it is no need to compute the exact SAD between this candidate block and the current block. It indicates that the search process can be terminated if the $\mathrm{SAD}^k(x,y)$ of a candidate block satisfies

$$\mathrm{SAD}^k(x,y) \ge \mathrm{SAD}(m,n) \quad \text{and} \quad k < N-1, \tag{6}$$

when we judge whether it is the best matched block or not, thus, the number of the searched pixels is less than the total number of pixels of this candidate block.

Obviously, to complete the motion estimation for a current block according to this idea, the number of the searched pixels is greatly less than the total number of pixels of all candidate blocks. Furthermore, no assumption is made on the matching criteria, so it will not exclude the optimum matched block. Therefore, according to this idea, an algorithm can be developed to reduce the computation burden of the matching process greatly without sacrificing the optimality of the solution.

The choice of the initial motion estimation affects the efficiency of this algorithm to some extent, good initial choice can terminate the matching process as soon as possible[9,10]. In this work, the motion vector of the corresponding block in the previous frame is used as the initial motion estimation for the current reference block.

To further reduce the search positions, it should be noted that the so far minimum matching error $\mathrm{SAD}(m,n)$ associated with the best matched block does not remain fixed during the search process. If at any step during the search, the $\mathrm{SAD}^{N-1}(x,y)$ calculated according to Eq. (4) for a candidate block with motion vector (x,y) is less than $\mathrm{SAD}(m,n)$, then $\mathrm{SAD}(m,n)$ is replaced by this $\mathrm{SAD}^{N-1}(x,y)$, and this block becomes the best matched block. This will progressively confine the search space and, hence, the computation of matching error between an ineligible candidate block and the current block can be terminated and this ineligible block can be eliminated as soon as possible. As a result, the computation burden is reduced drastically.

The flowchart of obtaining the optimal motion vector of a selected current block is shown in Fig. 1, and the detailed steps are depicted as follows:

Step 1) calculate the initial minimum SAD. As stating before, we select the candidate block with motion vector (0,0) in the previous frame as the best matched block, thus the so far minimum matching error SAD denoted as $\mathrm{SAD}_{\mathrm{so-far-min}}$ is SAD(0,0), that is

$$\mathrm{SAD}_{\mathrm{so-far-min}} = \mathrm{SAD}^{N-1}(0,0) = \sum_{j=0}^{N-1} \mathrm{SAD}_j(0,0). \tag{7}$$

Step 2) select an untested candidate block with motion vector (x,y).

Step 3) set $k=0$.

Step 4) calculate $\mathrm{SAD}^k(x,y)$ according to Eq. (4) for the selected candidate block with motion vector (x,y), and use $\mathrm{SAD}_{\mathrm{curr}}$ to represent $\mathrm{SAD}^k(x,y)$.

Step 5) if $\mathrm{SAD}_{\mathrm{curr}} < \mathrm{SAD}_{\mathrm{so-far-min}}$ and $k < N-1$, then $k = k+1$ and go to step 4). If $\mathrm{SAD}_{\mathrm{curr}} < \mathrm{SAD}_{\mathrm{so-far-min}}$ and $k = N-1$, then $\mathrm{SAD}_{\mathrm{so-far-min}} = \mathrm{SAD}_{\mathrm{curr}}$ and this block is the best matched block, and then go to step 6). If $\mathrm{SAD}_{\mathrm{curr}} \ge \mathrm{SAD}_{\mathrm{so-far-min}}$, then reject this candidate block, and terminate the computation of SAD for this block, and then go to step 6).

Step 6) if all candidate blocks have been tested, go to step 7). Otherwise, select another candidate block among the rest of untested candidate blocks in the search window, and go to step 3).

Step 7) get the best-matched block, and output vector (x,y) as the motion vector for this current block.

Theoretically, the proposed algorithm can reduce the computation burden obviously, for this algorithm can terminate the computation of matching evaluation that requires very intensive computation as early as possible and eliminate search positions in the search window as many as possible. In addition, during the search process, there is no assumption about the matching criteria, so this algorithm can guarantee to get the same optimal result as FSA.

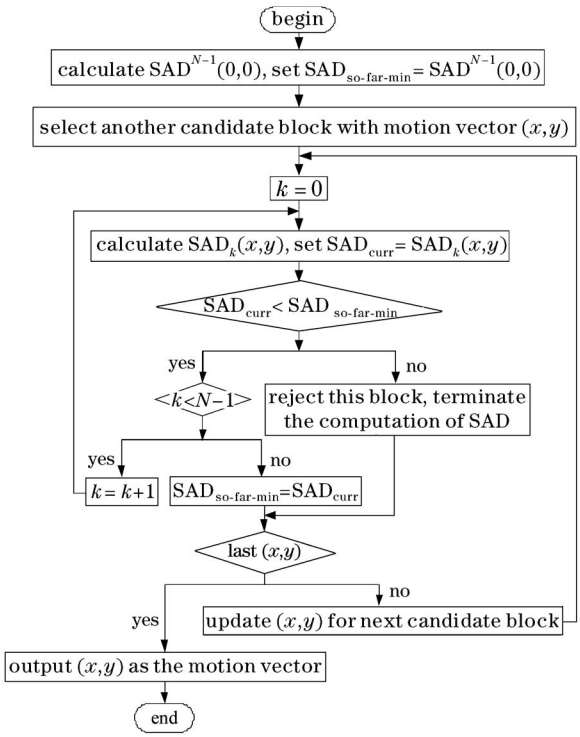

Fig. 1. Flowchart of obtaining the optimal motion vector of a selected current block.

Table 1. Performance Comparison of FSA and the Proposed Method

Image Sequence	Block Size	Average Exec. Time (s)			DFD		
		FSA	New Algorithm	TSS	FSA	New Algorithm	TSS
Coastguard	16 × 16	2.52	1.21	0.036	3.36	3.37	10.06
	8 × 8	0.71	0.43	0.042	3.29	3.29	12.46
Table Tennis	16 × 16	2.93	0.92	0.037	4.35	4.35	7.79
	8 × 8	0.72	0.30	0.046	4.36	4.36	7.33
Foreman	16 × 16	2.47	0.90	0.036	4.15	4.15	10.03
	8 × 8	0.71	0.28	0.042	4.57	4.57	11.60

Further more, some simulations have been performed to compare the performances of the proposed algorithms, FSA, and TSS, and three sequences are used in the simulations. The first is the coastguard sequence with size of 320 × 240. The second is the table tennis sequence with size of 352 × 240. The third is the forman sequence with size of 320 × 240. In order to evaluate the efficiencies of these three algorithms, two kinds of block sizes are adopted, one is 16 × 16, the other is 8 × 8, and the corresponding maximum allowed displacements are 16 and 8, respectively. Displaced frame difference (DFD)[11] is used to measure the performance of these algorithms and DFD is defined by

$$\frac{1}{WH}\sum_{i=1}^{W}\sum_{j=1}^{H}|(f(i,j,t)-f(i-u,j-v,t-1))|, \qquad (8)$$

where $f(i,j,t)$ represents the intensity of the pixel with coordinate (i,j) at frame t, (u,v) is the motion vector obtained at each point (i,j), and W and H are the image width and height, respectively.

As shown in the Table 1, the proposed algorithm can get the same accuracy as FSA with much more less computation load, for example, when the block size is 16×16, the computation load of the proposed algorithm is reduced by approximately 50% to 70%, and when the block size is 8 × 8, the computation burden of the proposed algorithm is reduced by approximately 40% to 60% of these three algorithms listed, TSS is the fastest one in view of speed, but the accuracy of TSS is the worst. In other word, when considering speed, the proposed algorithm is faster than FSA, but slower than TSS, and when considering accuracy, the proposed algorithm is the same as the FSA, and much better than TSS.

Also, the results show that the computation load of TSS do not change with image sequence when the block size is same, for the number of matching operations is equivalent. However, the results indicate that the computation burden of the proposed algorithm varies with image sequence, for different image sequences have the unique characteristics. It also implies that the computation loads of different blocks in the same image sequence are also different, which has been demonstrated by the simulation results. For example, among all blocks in a particular frame of foreman sequence, the minimum and maximum exec. time are 0.16 and 13.62 ms respectively when the block size is 16 × 16, and the minimum and maximum exec. time are 0.02 and 6.64 ms respectively when the block size is 8 × 8.

In this paper, an improved block matching technology for fast motion estimation has been developed. This improved algorithm tries to progressively confine the matching operation during the searching process, terminates unnecessary computation of matching error between the current block and the ineligible candidate block, and eliminates the ineligible block as early as possible. Still, the algorithm eliminates only blocks that with 100% certainty will be eliminated using FSA. Simulation results demonstrate that the estimation accuracy of the proposed algorithm is the same as that of FSA, but the computation burden has been reduced dramatically. When compared with TSS, the speed of this new algorithm is slower than that of TSS, but the accuracy is much better than that of TSS.

This work was supported by the National "973" Program of China (No. 2001CB309403) and the National Natural Science Foundation of China (No. 60574033). T. Tao's e-mail address is taotangfei@mail.xjtu.edu.cn.

References

1. C. Stiller and J. Konrad, IEEE Signal Proc. Magazine **16,** 70 (1999).
2. J. R. Jain and A. K. Jain, IEEE Trans. Communications **29,** 1799 (1981).
3. L.-P. Chau and X. Jing, in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing* **3,** 421 (2003).
4. S. Zhu and K.-K. Ma, IEEE Trans. Image Processing **9,** 287 (2000).
5. L. M. Po and W. C. Ma, IEEE Trans. Circuits Syst. Video Technol. **6,** 313 (1996).
6. T. Koga, K. Jinuma, A. Hirano, Y. Iijima, and T. Ishiguro, in *Proceedings of the National Telecommunications Conference (NTC)* G5.3.1 (1981).
7. Y. L. Chan and W. C. Siu, IEEE Trans. Circuits and Syst. Video Technol. **6,** 113 (1996).
8. S. Zafar, Y. Q. Zhang, and J. S. Baras, IEEE Trans. Broadcasting **37,** 97 (1991).
9. B. Liu and A. Zaccarin, IEEE Trans. Circuits and Syst. Video Technol. **3,** 148 (1993).
10. Y. Zhang and S. Mar, IEEE Trans. Broadcasting **37,** 102 (1991).
11. L.-W. Lee, J.-F. Wang, J.-Y. Lee, and J.-D. Shie, IEEE Trans. Circuits Syst. Video Technol. **3,** 85 (1993).