Genetic Block Matching Algorithm for Video Coding

Chun-Hung Lin
Commun. and Multimedia Lab.
Dept. of Comp. Sci. & Info. Eng.
National Taiwan University
Taipei, Taiwan, R. O. C.
d2506010@csie.ntu.edu.tw

Ja-Ling Wu
Commun. and Multimedia Lab.
Dept. of Comp. Sci. & Info. Eng.
National Taiwan University
Taipei, Taiwan, R. O. C.
wjl@cmlab.csie.ntu.edu.tw

Abstract

Genetic algorithms (GAs) recently have been successfully applied to perform block-based motion estimation. It is shown that the performance of the GA-based motion estimation algorithms nearly approaches that of the full search algorithm (FSA). However, the computational complexity of the existing GA-based algorithms is too high to be used in practice. In this paper, a lightweight genetic search algorithm (LGSA) is proposed. It can be seen from the simulation results that the performance of the proposed LGSA is not only as good as that of the FSA, but the computational complexity is also much lower than that of the FSA and the other existing genetic motion estimation algorithms.

1. Introduction

The block matching algorithms (BMAs) have been shown to be very efficient for the reduction of video bit rates [1]. It is therefore widely used in various kinds of video applications. In most of the video coding standards (such as H.261, MPEG-1, and MPEG-2), the block-based motion estimation technique plays a key role. Because of its importance, many works have been devoted to the development of fast searching algorithms. The FSA can find the optimal solutions (i.e., the motion vectors with the minimum matching errors) by exhaustively searching all possible blocks, but the tremendous computations make it difficult to be applied for real-time video compression, particularly for the software-based implementation. To meet the need of decreasing the computational complexity of the block matching process, different kinds of fast searching algorithms have been introduced [1]– [7]. Most of these fast algorithms have the restriction or the assumption that there should be only one minimum in the search space. Nevertheless, in practical applications, there are always many local minima in the search space, so it is very often for these algorithms to miss the optimal solution but to get the suboptimal one.

In the previous works [8, 9], GAs have been used to perform the block matching jobs. The results showed that GAs can solve the local minima sticking problem efficiently, hence they possess similar performance as that of the FSA. However, the necessary block matching number, which dominates the computational complexity of the motion estimation algorithms, is more than one-quarter of that of the FSA. Because the overhead for performing genetic operations is large, the computational complexity of these algorithms is almost similar to that of the FSA.

In this paper, a lightweight genetic block matching algorithm is proposed. The average block matching number of the proposed algorithm is very close to that of the three-step search algorithm (TSS), and its performance is very similar to that of the FSA. The control overheads are also improved while comparing to the previous works.

2. The Lightweight Genetic Block Matching Algorithm

Let S be a solution space and all the elements in S have their associated fitness values. The straight way to find the element with the maximum fitness value is to search among all the elements and to compare their fitness values. However, the computational complexity will be very high if the space size is large. In order to reduce the computational complexity, an efficient search algorithm should be applied.

If GAs are applied to search for the global maximum in S, a population P is maintained which consists of N elements, where N is the population size. Each el-

0-8186-7436-9/96 $5.00 © 1996 IEEE
Proceedings of MULTIMEDIA '96

ement in P is called a chromosome which is composed of a list of genes. The population P will evolve into another population P' by performing some genetic operations. The chromosomes with higher fitness values will have more probability to be kept in the population of the next generation, and to propagate their offspring. On the other hand, the weak chromosomes whose fitness values are small, will be replaced by another stronger chromosomes. Therefore, the quality of the chromosomes in the population will get better and better. After a suitable number of generations, the mature population is expected to contain the element with the global maximum value.

In this application, the solution space S is a set of motion vectors. The ith chromosome C_i in the population is defined as,

$$C_i = \begin{bmatrix} m_i \\ n_i \end{bmatrix} = \begin{bmatrix} a_{i,k-1} & a_{i,k-2} & \cdots & a_{i,1} & a_{i,0} \\ b_{i,k-1} & b_{i,k-2} & \cdots & b_{i,1} & b_{i,0} \end{bmatrix}, \quad 0 \le i \le N-1, \quad (1)$$

where the genes $a_{i,n}, b_{i,n} \in \{0, 1\}$, and $0 \le n \le k-1$. $[m_i\ n_i]^t$ represents one possible motion vector, and k is the codeword size of each motion offset whose value dependents on the search range. If the maximum motion offset is w, the value of k will be $\lceil \log_2 \frac{w}{2} \rceil$. The values of the genes are derived from the represented motion vector of the chromosome, that is,

$$a_{i,n} = \frac{m_i}{2^n} \ mod\ 2, \quad (2)$$

$$b_{i,n} = \frac{n_i}{2^n} \ mod\ 2, \quad (3)$$

where *mod* is the module operation.

Fig. 1 depicts the block diagram of the genetic evolution in the LGSA. An initial population is formed before the evolution. The initial chromosomes can be randomly selected from the search space, or be selected from the fixed locations of the search space.

Each chromosome has an associated fitness value which is defined as,

$$f_i = U_{\hat{d}_k - d_i} \cdot (\hat{d}_k - d_i) + \delta_{\hat{d}_k - d_i}, 0 \le i \le N-1, \quad (4)$$

where d_i is the matching error of the ith chromosome, and $\hat{d}_k$ is the kth minimum matching error among all the N values, $d_i, 0 \le i \le N-1$, and U and δ are a unit step function and a delta function, respectively. The constant k determines how many chromosomes could be selected at most as the seeds in the reproduction stage for producing a rival population. From (4), it is known that the chromosomes with smaller matching errors will have larger fitness values. The chromosomes with larger fitness values in the current population have larger probability to be selected as seeds of the next

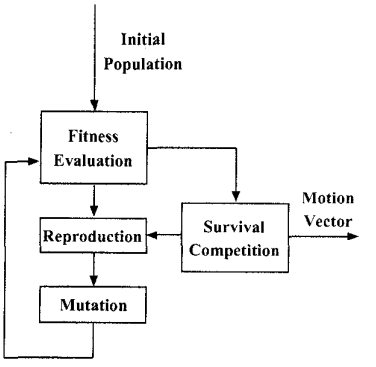

Figure 1. *The structural diagram of the genetic block matching algorithm.*

generation. This probabilistic schemes for selecting the seeds of new generations is known as the probabilistic reproduction.

The reproduction method used in this work is similar to the weighted roulette wheel method [10]. For each chromosome, an interval r_i is calculated as,

$$r_i = \left[\frac{\sum_{k=0}^{i-1} f_k}{\sum_{k=0}^{N-1} f_k}, \frac{\sum_{k=0}^{i} f_k}{\sum_{k=0}^{N-1} f_k} \right), 0 \le i \le N-1, \quad (5)$$

where f_k is the fitness value of the kth chromosome in the population, and '[' and ')' denote closing and opening boundaries, respectively. When the interval of each chromosome has been determined, N real numbers, α_i, are randomly generated, where $0 \le \alpha_i < 1$ and $0 \le i \le N-1$. The value of α_i will be bounded by some r_j, that is, $\alpha_i \in r_j$. The chromosome C_j is then selected as a seed to generate the rival population. It is possible that one chromosome can be selected twice or more. Because N real random numbers are generated, N seeds could be selected and placed in the mating pool.

After the reproduction stage, each seed in the mating pool will be processed and transferred into a candidate chromosome of the new generation. Assume the current seed to be processed is $[m_i\ n_i]^t$, where $m_i = [a_{i,k-1}a_{i,k-2}\ldots a_{i,0}]$ and $n_i = [b_{i,k-1}b_{i,k-2}\ldots b_{i,0}]$. In the jth generation, the two genes $a_{i,z}$ and $b_{i,z}$ are varied, where $z = k-1-j$. There are eight mutation operators, $\{(\zeta_p, \eta_p) | 0 \le p \le 7\}$, which can be applied in our implementation, that is,

$$a'_{i,z} = a_{i,z} + \zeta_p, \quad (6)$$

$$b'_{i,z} = b_{i,z} + \eta_p, \quad (7)$$

where p is a random integer number whose value is between 0 and 7. Because the chromosomes are randomly

selected and put on the mating pool, it is not necessary to generate a random number for determining the p's value. We simply set p to be $(i \ mod \ 8)$. The mutation operations are defined as,

$$\zeta_p = (-1)^m([p+1-m(m+1)] \cdot [1-(\lfloor 2\sqrt{p+1} \rfloor \ mod \ 2)] + \lceil \frac{1}{2}m \rceil), \quad (8)$$

$$\eta_p = (-1)^m([p+1-m(m+1)] \cdot [\lfloor 2\sqrt{p+1} \rfloor \ mod \ 2] - \lceil \frac{1}{2}m \rceil), \quad (9)$$

$$m = \lfloor \sqrt{p+1} \rfloor. \quad (10)$$

When the mutation operations are performed on the most significant genes of the chromosomes (e.g., a_{k-1}, b_{k-1}, etc.), the chromosomes which are far from the original ones in the search space are generated. Whereas, the nearby chromosomes are generated when the mutation operations are performed on the least significant genes.

There are N chromosomes in the mating pool after performing the genetic operations. Along with the original chromosomes in the current generation, N chromosomes are selected from these $2N$ chromosomes according to their fitness values. Each chromosome can only be selected at most once. The chromosomes with larger fitness values will be picked up as the members of the population in the next generation, and go through the next iterations of the genetic evolution. Although the sorting operation is needed in this survival competition stage, the overhead is not high because the population size is usually not large in this application. This stage is added to the proposed algorithm to prevent the chromosomes from being destroyed by the new ones with poorer fitness values, because the new chromosomes generated from the original ones are not guaranteed to have larger fitness values in GAs.

The chromosome with the maximum fitness value is selected from the current population as the possible solution. The possible solution might be replaced by the others from one generation to the other generations. The iteration will be terminated if the matching error of the solution is less than a predefined threshold, or the iteration number is equal to k — the codeword size of the chromosomes.

3. Experimental Results

The proposed LGSA along with the FSA and the TSS were implemented[1] to compare their performance. In the simulations, the block size for block matching is 8×8 pixels, and the maximum motion offset is 16 pixels. The TSS was directly expanded to four steps to cope with the search range.

Tables 1 and 2 list the average performance of the above three search methods both in the cases of the normal frame rate (30 frame/sec) and the lower frame rate (10 frame/sec). It can be seen from the table that the LGSA achieves similar performance to the FSA. When the frame rate is low, the TSS method is shown to have poor performance because of the increase of the local minima in the search space. Nevertheless, the LGSA still have the similar performance to that of the FSA.

Algorithm	PSNR(dB)					
	sales	miss	claire	trevor	football	tennis
FSA	36.33	38.62	42.66	35.09	26.02	26.94
TSS	35.75	38.01	41.85	34.13	24.12	25.76
LGSA	36.25	38.38	42.61	34.93	25.27	26.40

Table 1. *Comparisons of the average performance in normal frame rate.*

Algorithm	PSNR(dB)					
	sales	miss	claire	trevor	football	tennis
FSA	34.83	37.69	39.07	33.58	23.60	24.44
TSS	33.85	35.57	36.79	31.81	22.02	23.03
LGSA	34.59	37.09	38.69	33.09	22.57	23.63

Table 2. *Comparisons of the average performance in low frame rate.*

In the low frame rate case, the subjective quality of the motion compensated images generated by the TSS are unacceptable. Whereas, the motion compensated images generated by the LGSA are still good enough. When the allowed bit-rate is limited, the decoded images which are motion estimated by the LGSA will have acceptable quality.

To compare the computational complexity, Table 3 shows the average numbers of the searching points needed for each block in different algorithms. From the table, it is known that the LGSA needs similar searching number as that of the TSS. However, the overheads of the genetic operations (reproduction, mutation, and survival competition, etc.) must be taken into account. Fig. 2(b) shows the ratio diagram of the average execution time for these algorithms. All the algorithms are run on a SUN SPARC-10 workstation. It follows from the figure that the LGSA needs more computa-

[1]The source codes of the LGSA can be found on "http://www.cmlab.csie.ntu.edu.tw/~d2506010/pub/lgsa/lgsa.html". Some of the motion compensated images can also be found there for comparing the perceptual quality of different search algorithms.

tions than the TSS does, but still has fairly low computational complexity as compared to the FSA. When the frame rate is high and the video sequences contain only slow moving objects, the computational costs of the genetic method become non-negligible. This is because the performance of the TSS method is only a little worse than that of the FSA, i.e., the improvement achieved by the genetic method is very limited. On the other hand, the computational complexity of the LGSA becomes reasonable when the frame rate is low or when the objects in the video sequences move very fast, because the performance of the TSS is unacceptable in these cases.

Algorithm	FSA	TSS	LGSA
Average Search Number	1010.45	31.74	51.51

Table 3. *Average search number for each block.*

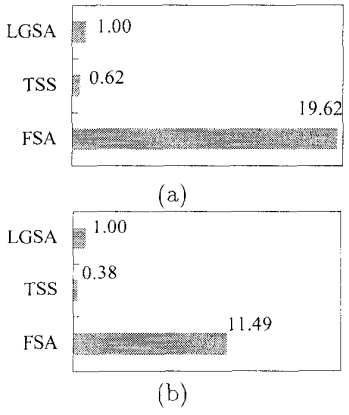

Figure 2. *Comparisons of the computational complexity ratio, (a) average search number, (b) average execution time.*

4. Conclusion

In this paper, a new GA-based block matching algorithm is proposed. By applying GAs, the proposed algorithm overcomes the local minima sticking problem by including several search points simultaneously during the searching process. It is shown that the performance is very similar to the FSA. Due to the special design of the genetic operations, the computational complexity is under control (i.e., very close to the TSS) and is much lower than that of the FSA. In low bit-rate video coding where the frame rate is not high or in the applications where the motion of objects is violent, the LGSA can be applied without much degrading of the image quality.

References

[1] J. R. Jain and A. K. Jain. Displacement measurement and its application in interframe image coding. *IEEE Trans. Commun.*, (1):1799–1808, Dec. 1981.

[2] R. Srinivasan and K. R. Rao. Predictive coding based on efficient motion estimation. *IEEE Trans. Commun.*, pages 888–895, Aug. 1985.

[3] C. H. Hsich, P. C. Lu, J. S. Shyn and E. H. Lu. Motion estimation algorithm using interblock correlation. *IEE Electron. Lett.*, (5):276–277, Mar. 1990.

[4] A. Puri, H. M. Hang and D. L. Schiling. An efficient block-matching algorithm for motion compensated coding. In *Proc. IEEE ICASSP'87*, pages 25.4.1–25.4.4, 1987.

[5] M. Ghanbari. The cross-search algorithm for motion estimation. *IEEE Trans. Commun.*, pages 950–953, July 1990.

[6] T. Koga, K. Iinuma, A. Hirano, Y. Iijima and T. Ishiguro. Motion compensated interframe coding for video conferencing. In *Proc. Nat. Telecommun. Conf.*, pages 5.3.1–5.3.5, New Orleans, LA, Nov. 1981.

[7] H-M. Jong, L-G. Chen and T-D. Chiueh. Accuracy improvement and cost reduction of 3-step search block matching algorithm for video coding. *IEEE Trans. Circuits Syst. Video Technol.*, (1):88–90, Feb. 1994.

[8] Keith Hung-Kei Chow and Ming L. Liou. Genetic motion search algorithm for video compression. *IEEE Trans. Circuits Syst. Video Technol.*, (6):440–445, Dec. 1993.

[9] In Kwon Kim and Rae-Hong Park. Block matching algorithm using a genetic algorithm. In *in SPIE Symposium on Visual Communications and Image Processing*, pages 1545–1552, Taipei, Taiwan, May 1995.

[10] J. H. Holland. *Adaptation in Natural and Artificial Systems.* University Michigan Press, AnnArbor, 1975.