The Cross-Search Algorithm for Motion Estimation

M. GHANBARI

Abstract—**A fast block matching algorithm for motion estimation is presented. It is based on a logarithmic step where in each search step only 4 locations are tested. For a motion displacement of w pels/frame, this technique requires $5 + 4 \log_2 w$ computations to locate the best match. Using sequences of CIF standard pictures, the interframe motion compensated prediction error with this technique is compared to the other fast methods. The computational complexity of this algorithm is also compared against those methods.**

I. Introduction

INTERFRAME predictive coding is one of the most powerful image coding techniques which can eliminate redundancy in natural scenes. In conventional interframe predictive coding the difference between the current picture element (pel) and its prediction value from the previous frame is coded and transmitted. At the receiver the error signal is added to a similar prediction value to reconstruct the picture. The better the predictor, the smaller the error signal, and hence the transmission bit rate. If the scene is still, a good prediction for the current pel is the same pel in the previous frame. However, when there is motion, assuming that movement in the picture is only a shift of object position, then a pel on the same part of the moving object becomes a better prediction.

Application of the knowledge of the displacement of a moving object in successive frames is called motion compensation. Various motion compensation algorithms have been applied to interframe predictive coding [1]. One of these, which is the subject of this paper, is known as the block matching algorithm (BMA), which estimates the amount of motion on a block by block basis. This algorithm, and various fast search methods that reduce the necessary computations, are reviewed shortly. A new fast search method named the "cross-search algorithm" (CSA) is presented and its computational complexity and performance is compared to the other methods.

II. Block Matching Algorithm (BMA)

In a typical BMA, a frame is divided into blocks of $M \times N$ pels or, more favorably, square blocks of N^2 pels [2]. Then for a maximum motion displacement of w pels per frame, the current block of pels is matched against a corresponding block at the same coordinate but in the previous frame, within the square window of width $N + 2w$ (Fig. 1).

The best match on the basis of a matching criterion yields the displacement.

Various measures such as the cross-correlation function (CCF), mean squared error (MSE), and mean absolute error (MAE) can be used in the matching criterion [2]–[4]. For the best match, in the CCF the correlation has to be maximized, whereas in the latter two the distortion must be minimized. In the experiment, both MSE and MAE were used, since it was believed CCF would not give good motion tracking, especially when the displacement was not large [4].

Paper approved by the Editor for Image Communication Systems of the IEEE Communications Society. Manuscript received February 17, 1988; revised April 18, 1989. This paper was presented at 1988 Picture Coding Symposium, Torino, Italy, September 12–14, 1988.

The author is with the Department of Electronic Systems Engineering, University of Essex, Colchester C04 3SQ, England.

IEEE Log Number 9036317.

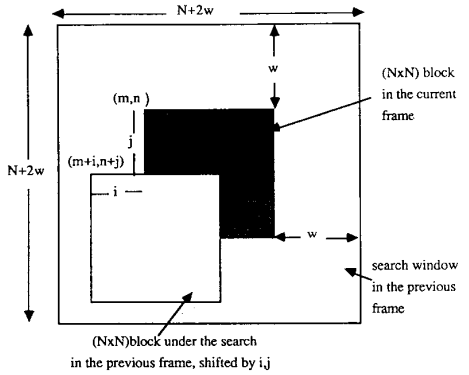


Fig. 1. The current and previous frames in a search window.

The matching functions of the type MSE and MAE are defined as for MSE;

$$M(i,j)=\frac{1}{N^2}\sum_{m=1}^{N}\sum_{n=1}^{N}(f(m,n)-f(m+i,n+j))^2 \qquad -w\le i,j\le w \quad (1)$$

for MAE;

$$M(i,j)=\frac{1}{N^2}\sum_{m=1}^{N}\sum_{n=1}^{N}|f(m,n)-f(m+i,n+j)| \qquad -w\le i,j\le w \quad (2)$$

where $f(m, n)$ represents the current block of N^2 pels at coordinates (m, n) and $f(m + i, n + j)$ represents the corresponding block in the previous frame at new coordinates $(m + i, n + j)$.

To locate the best match by full search, $(2w + 1)^2$ evaluations of the matching criterion are required. To reduce the computational complexity, Jain and Jain [4] use a two-dimensional logarithmic search method (TDL) to track the direction of a minimum mean squared error distortion measure. In their method, the distortion for the five initial positions, one at the center of the coordinate and four at coordinates $(\pm w/2, \pm w/2)$ of the search window, are computed first. In the next step, three more positions with the same step size in the direction of the previous minimum position are searched. The step size is then halved and the above procedure is continued until the step size becomes 1. Finally all the nine positions are searched. With this method, for $w = 5$ pels/frame, 21 positions are searched as opposed to 121 positions required in the full search method.

Koga *et al.* [5] use a three-step motion vector direction search (TSS) to compute displacements up to 6 pels/frame. In their method all eight positions surrounding the coordinate with a step size of $w/2$ are searched first. At each minimum position the search step size is halved and the next eight new positions are searched. This method, for $w = 6$ pels/frame, searches 25 positions to locate the best match.

In Kappagantula and Rao's [2] modified motion estimation algorithm (MMEA), prior to halving the step sizes, two more positions are also searched. With this method for $w = 7$ pels/frame,

0090-6778/90/0700-0950$01.00

only 19 MAE computations are required. In Srinivasan and Rao's [6] conjugate direction search (CDS) method, at every iteration of the direction search, two conjugate directions with a step size of one pel, centered at the minimum position are searched. Thus, for $w = 5$ pels/frame, there will be only 13 searches at most.

Recently, Puri *et al.* [7] have introduced the orthogonal search algorithm (OSA) where with a logarithmic step size, at each iteration 4 new locations are searched. In this method, at every step, 2 positions are searched alternately in the vertical and horizontal directions. The total number of test points is $1 + 4 \log_2 w$.

III. Cross-Search Algorithm (CSA)

In the cross search algorithm presented here, the basic idea is still a logarithmic step search, which has also been exploited in [2], [4], [5], but with some differences which lead to fewer computational search points. The main difference between the presented method and the other logarithmic reduced search methods is that in this method, at each iteration there are 4 search locations, which are the end points of a cross (X) rather than (+). Also, at the final stage, the search points can be either the end points of (X) or (+) crosses, as described below.

The reference point for a block is taken to be the pel(i, j) at its upper left hand corner. The block in the previous frame which corresponds to a block in the current frame is referred to as the block at (0,0). The cross search algorithm can then be described as follows.

Step 1: The current block and the block at (0,0), are compared and if the value of the distortion function is less than a predefined threshold T then the current block is classified as a nonmoving block and the search stops. Otherwise go to Step 2.

Step 2: Initialize the minimum position (m, n) at $m = 0, n = 0$ and set the search step size p equal to half of the maximum motion displacement w, i.e., $p = w/2$.

Step 3: Move the coordinates (i, j) to the minimum position (m, n), that is $i = m$ and $j = n$.

Step 4: Find the minimum position (m, n) of the coordinates (i, j), $(i - p, j - p)$, $(i - p, j + p)$, $(i + p, j - p)$ and $(i + p, j + p)$.

Step 5: If $p = 1$ go to Step 6, otherwise halve the step size p, then go to Step 3.

Step 6: If the final minimum position (m, n) is either (i, j), $(i - 1, j - 1)$ or $(i + 1, j + 1)$ go to Step 7, otherwise go to Step 8.

Step 7: Search for the minimum position at (m, n), $(m - 1, n)$, $(m, n - 1)$, $(m + 1, n)$ and $(m, n + 1)$. Here the end points of a Greek cross (+) are searched.

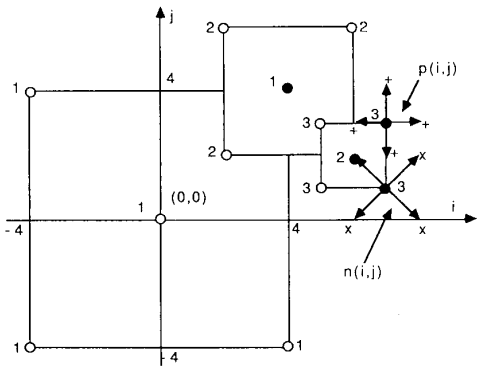
Step 8: Search for the minimum position at (m, n), $(m - 1, n - 1)$, $(m - 1, n + 1)$, $(m + 1, n - 1)$ and $(m + 1, n + 1)$. In this case the end points of a St. Andrew's cross (X) are searched.

An example of the cross search algorithm for a maximum motion displacement of $w = 8$ pels/frame is shown in Fig. 2. As can be seen, there are four new positions to be searched in each stage and four extra positions to be searched in the final stage. In addition to this there is the calculation for motion detection at point (0,0). Thus, for the maximum motion displacement of w pels/frame, the total number of computations becomes $5 + 4 \log_2 w$.

IV. Computational Efficiency of the Cross-Search Algorithm (CSA)

To perform a direct comparison among the computational efficiencies of the CSA and of the other fast search methods, the computational complexity of each method has been formulated for a motion displacement of size w. This is because the figures given for these methods are based on different motion displacement values, and so direct comparison is impossible. Table I shows the computational complexities of the methods when w is a power of 2.

As can be seen from Table I, the orthogonal search algorithm is the fastest search method. For motion displacements up to 4 pels/frame, the conjugate direction search method is the second most efficient algorithm, but it becomes inefficient when w is greater than 4 pels/frame. The efficiency of the CSA for values of w above 8 pels/frame comes in second place.



White circles denote the search step.

Black circles denote the minimum position at each step.

p(i,j) is the final minimum position for the Greek cross search.

n(i,j) is the final minimum position for the St. Andrew's cross search.

+ are the pels searched with the Greek cross.

x are the pels searched with the St. Andrew's cross.

Fig. 2. An example of the CSA search for $w = 8$ pels/frame.

TABLE I
COMPUTATIONAL COMPLEXITY OF THE SEARCH ALGORITHMS

Algorithm	Maximum number of search points	w		
		4	8	16
FSM	$(2w+1)^2$	81	289	1089
TDL [4]	$2 + 7 \log_2 w$	16	23	30
TSS [5]	$1 + 8 \log_2 w$	17	25	33
MMEA [2]	$1 + 6 \log_2 w$	13	19	25
CDS [6]	$3 + 2 w$	11	19	35
OSA [7]	$1 + 4 \log_2 w$	9	13	17
CSA	$5 + 4 \log_2 w$	13	17	21

Some logarithmic searches may not be able to search all the pel positions at the boundaries of the search window. In the two-dimensional logarithmic search (TDL) some pels at the corner of the search window are not searched. In CSA almost 25% (one in every four) of the pels at the boundaries are not searched. In MMEA, TSS and OSA none of the pels at the boundaries are searched. In fact the optimum motion displacement length for these methods is $w - 1$, thus the computational complexity of these methods should be accounted for the lengths 3, 7, and 15.

V. Performance Efficiency of the Cross-Search Algorithm

The efficiency of motion estimation algorithms is often measured in terms of the entropy performance of the motion compensated prediction error. In the experiment, two CIF standard (352 pels × 288 lines, 30 frames/s and 8 bits/pel) picture sequences, "split screen" and "Trevor White," each of 60 frames were used as source material.

An interframe predictive coder, but without the quantizer in the prediction loop, was simulated. The reason for excluding the quantizer was to eliminate any quantization distortion in the prediction error. First of all, tests were made to choose the best matching function between the mean squared and mean absolute error distortion functions. Using the motion estimation algorithm of the

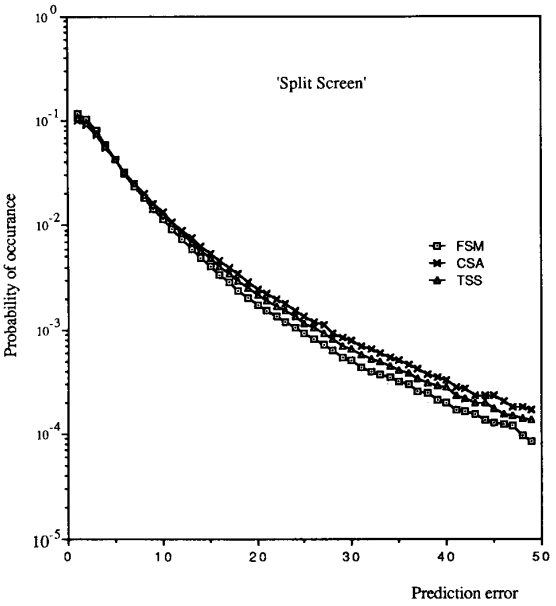


Fig. 3. The probability density functions of the motion compensated errors for the "split screen" sequence by either of the methods.

□ full search method (FSM)
△ three-step motion vector direction search (TSS) [5]
X cross-search algorithm (CSA)

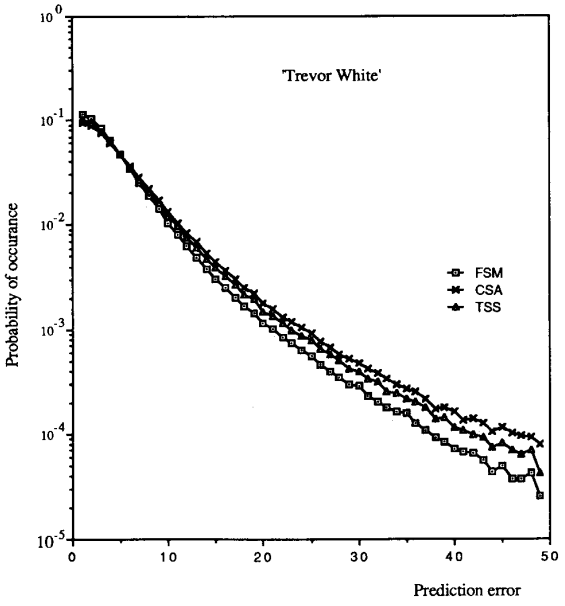


Fig. 4. The probability density functions of the motion compensated errors for the "Trevor White" sequence by either of the methods.

□ full search method (FSM)
△ three-step motion vector direction search (TSS) [5]
X cross-search algorithm (CSA)

proposed type (CSA), the stationary blocks of 8 × 8 pels were detected whenever the average absolute value of the difference between the current block and the corresponding block in the previous frame was less than 4 out of 256. For moving blocks the best match was found on the basis of both MSE and MAE matching functions using (1) and (2), respectively. The entropies of the motion compensated prediction errors and their standard deviations for both "split screen," and "Trevor White" sequences show that the MAE results in marginally smaller prediction error than the MSE (almost 0.8% of the bit rate) even though its standard deviation is slightly higher (by the same amount).

Since the MAE distortion functions gives better motion tracking than the MSE, the former was used in the matching criterion of the algorithms to be compared. Here, for the moving blocks of 8 × 8 pels, the motion compensated prediction errors of all methods as well as those derived by the full search method (FSM), for both picture sequences were recorded. Figs. 3 and 4 compare the performance of the CSA to the FSM and the TSS, the most efficient reduced search method, for the split screen and Trevor White sequences, respectively. As can be seen both fast methods, CSA and TSS, closely follow the full search, but TSS has a better performance.

The entropies and standard deviations of the motion compensated prediction errors for all the methods are tabulated in Table II. In this table, all the reduced search methods have comparable performance with that of the FSM. Referring to both Tables I and II, it can be concluded that increasing the number of search points, also increases the motion compensation efficiency. It can be seen that, the cross search algorithm, which possesses the second lowest computational complexity, performs better than some of the other algorithms with higher computational complexities.

TABLE II
ENTROPIES AND STANDARD DEVIATIONS OF THE PREDICTION ERRORS FOR "SPLIT SCREEN" AND "TREVOR WHITE" SEQUENCES

Algorithm	Split Screen		Trevor White	
	Entropy (bits/pel)	Standard deviation	Entropy (bits/pel)	Standard deviation
FSM	4.57	7.39	4.41	6.07
TDL	4.74	8.23	4.60	6.92
TSS	4.74	8.19	4.58	6.86
MMEA	4.81	8.56	4.69	7.46
CDS	4.84	8.86	4.74	7.54
OSA	4.85	8.81	4.72	7.51
CSA	4.82	8.65	4.68	7.42

VI. Conclusions

A fast search block matching algorithm for motion estimation and prediction has been presented. For a maximum motion displacement of w pels/frame, this algorithm requires only $(5 + 4 \log_2 w)$ computations to locate the best match. The interframe prediction error of this method, for a maximum motion displacement of $w = 8$ pels/frame is 0.25 to 0.27 bits/pel inferior to the full search method, however, its computational complexity is 17 times less than the latter.

In comparison with the other fast block matching algorithms, while the computational complexity of the presented method is the second lowest, its compensation performance is not. For motion displacement of $w = 8$ pels/frame, it is 25/17 times faster than the most efficient reduced search method; the three-step vector motion detection [5], whereas its performance is 0.08 bits/pel poorer.

Finally, it is worthy of note that the use of mean absolute error as the distortion measure, results in slightly better entropy performance than with the more complex mean square error (almost 0.8% lower prediction error).

REFERENCES

[1] T. Ishiguro and K. Iinuma, "Television bandwidth compression transmission by motion-compensated interframe coding," *IEEE Commun. Mag.*, vol. 10, pp. 24–30, 1982.

[2] S. Kappagantula and K. R. Rao, "Motion compensated predictive coding," in *Proc. Int. Tech. Symp. SPIE*, San Diego, CA, Aug. 1983.

[3] H. C. Bergmann, "Displacement estimation based on the correlation of image segments," *IRE Conf. Electron. Image Processing*, York, U.K., July 1982.

[4] J. R. Jain and A. K. Jain, "Displacement measurement and its application in interframe image coding," *IEEE Trans. Commun.*, vol. COM-29, pp. 1799–1808, Dec. 1981.

[5] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, "Motion compensated interframe coding for video conferencing," in *Proc. Nat. Telecommun. Conf.*, New Orleans, LA, Nov. 29–Dec. 3, 1981, pp. G5.3.1–5.3.5.

[6] R. Srinivasan and K. R. Rao, "Predictive coding based on efficient motion estimation," *IEEE Int. Conf. Commun.*, Amsterdam, May 14–17, 1984, pp. 521–526.

[7] A. Puri, H. M. Hang, and D. L. Schilling, "An efficient block-matching algorithm for motion compensated coding," *Proc. IEEE ICASSP 1987*, pp. 25.4.1–25.4.4.