Fast motion estimation with search-center prediction

Hing Yip Chung
N. H. C. Yung
P. Y. S. Cheung
The University of Hong Kong
Department of Electrical & Electronic Engineering
Pokfulam Road
Hong Kong SAR
E-mail: hychung@eee.hku.hk

Abstract. This paper presents a new block-based motion estimation algorithm that employs motion-vector prediction to locate an initial search point, which is called a search center, and an outward spiral search pattern with motion-vector refinement, to speed up the motion estimation process. It is found that the proposed algorithm is only slightly slower than cross search, but has a peak signal-to-noise ratio (PSNR) very close to that of full search (FS). Our research shows the motion vector of a target block can be predicted from the motion vectors of its neighboring blocks. The predicted motion vector can be used to locate a search center in the search window. This approach has two distinct merits. First, as the search center is closer to the optimum motion vector, the possibility of finding it is substantially higher. Second, it takes many less search points to achieve this. Results show that the proposed algorithm can achieve 99.7% to 100% of the average PSNR of FS, while it only requires 1.40% to 4.07% of the computation time of FS. When compared with six other fast motion estimation algorithms, it offers the best trade-off between two objective measures: average PSNR and search time.
© *2001 Society of Photo-Optical Instrumentation Engineers.* [DOI: 10.1117/1.1367865]

Subject terms: block matching; motion estimation; search center; motion vector prediction; search pattern.

Paper 200014 received Jan. 11, 2000; revised manuscript received July 24, 2000 and Dec. 14, 2000; accepted for publication Dec. 27, 2000.

1 Introduction

Motion estimation has been a hot research topic for years. It is the most important part of video compression and coding, as it exploits as much temporal redundancy as possible to reduce the size of the data required in digital video storage and transmission. Low-bit-rate video transmission is therefore impossible without the use of motion estimation. Although motion estimation is such a useful method in reducing the size of a coded video sequence, it is computationally intensive, which makes real-time video coding, though not impossible, a difficult task. Parallelization may help, but motion estimation often lies on the critical path. In a typical video encoding system, motion estimation (full-search block matching) can take 50% (Ref. 1) to 75% (Ref. 2) of the computation time.

In the past two decades, there has been extensive research into motion estimation techniques. Many such techniques, including pel-recursive techniques,[3–5] gradient techniques,[6–8] frequency-domain techniques,[9,10] and block-based matching techniques, have evolved. Among these, block-based matching has been widely adopted for international standards such as the H.261,[11] H.263,[12] MPEG-1,[13] and MPEG-2,[14] due to its effectiveness and robustness. Therefore, most of the research work has been concentrated on optimizing the block-based motion estimation technique.

As the demand for real-time video coding increases for different applications (video recording, video conferencing, video phone, etc.), fast video encoding with good compression ratio as well as high signal-to-noise ratio is essential. Good compression ratio means reducing the size of the coded video with graceful degradation of quality. Motion estimation is a technique designed exactly to achieve good compression ratio in video compression. However, speed and quality are often conflicting goals. Nowadays, researchers are still actively seeking an optimum trade-off between these two factors.

Most of the motion estimation algorithms proposed tend to be biased towards achieving speed by sacrificing visual quality. In view of this, we were motivated to find a good trade-off between speed and quality, that is, to increase the speed as much as is consistent with good visual results. We focused on the block-based motion estimation technique, since it is widely adopted in international standards.

In this paper we propose a model to formulate a method to predict the search center (initial probe point in the search space) by using the spatial information in the current frame, to reduce the search space for motion estimation. A search pattern biased towards the search center can also help in reducing the search space. Thus, a search-center-biased search pattern can speed up the searching process in motion estimation. In general, the proposed algorithm has a peak signal-to-noise ratio (PSNR) very close to that of full search (FS) with substantial speedup. Four sequences, ‘‘Susie,’’ ‘‘Football,’’ ‘‘Flower Garden,’’ and ‘‘Mobile and Calendar,’’ were used to test the proposed method. From our results, it is found that our method is able to achieve 99.7% to 100% of the average PSNR of FS while only requiring 1.40% to 4.07% of the computation time. In comparison with three-step search (TSS), one of the most popular motion estimation algorithms, our proposed method can achieve better quality (0.8% to 4% better) while requiring 39.3% to 55% less computation.

2 Block-Based Motion Estimation

2.1 *Overview*

The principle of block-based motion estimation in most of the video standards is that the video image frame is partitioned into blocks, and each block is an elementary unit. Motion estimation is performed by matching each block in the current frame against a region in a reference frame to find the best match. The criterion for the best match is well accepted to be the minimum error, or energy, of the residue obtained from the subtraction of corresponding pixels between the blocks, which is given by the following equation[15]:

$$E(B,\mathbf{v})=\sum_{\mathbf{r}\in B}|I_{\mathrm{cur}}(\mathbf{r})-I_{\mathrm{ref}}(\mathbf{r}+\mathbf{v})|^n, \tag{1}$$

where B contains all the pixels in the current block, $\mathbf{v}$ is a displacement vector that addresses the reference block location in I_{ref}, $I_{\mathrm{cur}}(\mathbf{r})$ is the intensity of the pixel at $\mathbf{r}$ of the current frame I_{cur}, and $I_{\mathrm{ref}}(\mathbf{r})$ is the intensity of the pixel at $\mathbf{r}$ of the reference frame I_{ref}. When $n=1$, Eq. (1), evaluates the sum of absolute differences (SAD), and when $n=2$, it evaluates the total energy instead. It is more sensible to have $n=2$, since if the total energy is lower, the number of bits required to code the residue is smaller. However, n is usually set to 1 in practice, since that involves no multiplication and hence lower computation cost. In particular, when $E(B,\mathbf{v})$ is divided by the total number of pixels in the block, the resultant quantity will be the mean absolute error (MAE) or mean squared error (MSE).[15] By using one of these matching criteria, the best match can be located. Then the motion vector (MV) obtained for the block B can be generally formulated as follows:

$$\mathbf{MV}(B)=\arg\min_{v\in S}E(B,\mathbf{v}), \tag{2}$$

where S is the search area, which consists of all possible motion vectors.

$\mathbf{MV}(B)$ is well accepted to be the optimal solution because it is the motion vector that yields the lowest MAE or MSE, which in turns result in the highest PSNR. One obvious way of finding $\mathbf{MV}(B)$ is to do an exhaustive search, i.e., the FS algorithm. Although FS is computationally intensive, it can always guarantee an optimum solution to Eq. (2). In view of this, many researchers have tried to find ways of cutting corners in order to speed up the process of finding the desired MV. A lot of fast motion estimation algorithms have been proposed.[16–35] Most of them have successfully reduced the computation complexity of finding the MV, but few of them can guarantee an optimum solution. These proposed algorithms can be generally classified into two categories: fast matching and fast search algorithms.

2.2 *Fast Matching Algorithms*

Fast matching methods use a matching criterion $E'(B,\mathbf{v})$ other than $E(B,\mathbf{v})$ in Eq. (1) for finding a best match. The reason is that some pixels in a block contribute most of the error or energy to the residue. Therefore, it is believed that not all the pixels are needed in the matching criterion. Thus, fast matching methods use another matching criterion derived from a subset of pixels:

$$E'(B,\mathbf{v})=\sum_{\mathbf{r}\in B'}|I_{\mathrm{cur}}(\mathbf{r})-I_{\mathrm{ref}}(\mathbf{r}+\mathbf{v})|^n, \quad \text{where} \quad B'\subset B. \tag{3}$$

The selection of the pixels in B' can be either static[16–18] or dynamic.[19–21] Standard subsampling[16–17] is an example of static pixel selection. Some other algorithms select pixels that possess special features, such as the edge pixels in the block[19–21] or those with the largest gradient magnitude.[22] These constitute dynamic pixel selection. Both static and dynamic selection can reduce the number of pixels needed for evaluating the matching function. The amount of computation saved can be increased by adjusting the subsampling ratio or by limiting the number of pixels to be selected from the block, and hence can be chosen to suit the problem. However, subsampling can lead to severe degradation in quality, since the error contributed by the discarded pixels may be high. Dynamic selection of pixels is better at preserving the quality, though at the cost of more computation, since preprocessing (such as edge detection or gradient evaluation) must be done before the selection process can begin.

2.3 *Fast Search Algorithms*

Fast search methods do not modify the matching criterion $E(B,\mathbf{v})$ as such. They speed up the search by reducing the number of search points in the search area. This basically reduces the search space of the whole searching process. Fast search can be described by the following equation:

$$\mathbf{MV}'(B)=\arg\min_{\mathbf{v}\in S'}E(B,\mathbf{v}), \quad \text{where} \quad S'\subset S. \tag{4}$$

When S' is a constant set, it means that the search area S has been subsampled by a constant pattern. When S' is a dynamic set, it can be determined by the MV's of the neighboring blocks, or the MVs of the blocks in the previous frames. Some fast search algorithms find the MVs in iterative steps where the search space is determined by the previous iteration. They can be formulated as follows:

$$\mathbf{MV}^n(B)=\arg\min_{\mathbf{v}\in S'(\mathbf{MV}^{n-1}(B),n)}E(B,\mathbf{v}), \quad \text{where} \quad S'\subset S. \tag{5}$$

In the above equation, $S'(\cdot,\cdot)$ is a function of the current iteration number and the MV of the previous iteration. Some fast search algorithms set the number of iterations to a constant, as in the TSS,[23] cross search CS,[24] and one-dimensional FS (1DFS).[25] Some make it dynamic and just iterate until a termination rule is satisfied. One-at-a-time search (OTS),[26] new three-step search (NTSS),[27] four-step search (FSS),[28] and diamond search (DS)[29] are examples of this kind of fast search algorithms. Among all these algorithms, TSS is the most widely used due to its simplicity.

Fast search algorithms like those mentioned above can reduce the computation cost a great deal. The reduction in the number of search points is usually justified by the assumption that the matching function $E(B,\mathbf{v})$ (or the MAE or MSE) increases monotonically when the distance between the search points and the absolute minimum, $|\mathbf{v}-\mathrm{MV}(B)|$, increases. By using this assumption, many other fast search algorithms[30,31] successfully reduce the number of search points by using different S'. However, when $E(B,\mathbf{v})$ is plotted on the plane spanned by $\mathbf{v}$, the surface is neither unimodal nor smooth. Although S' varies among these algorithms, they all suffer from the possibility of being trapped in a local minimum, resulting in nonoptimal solutions.

On the other hand, some fast search algorithms make use of the temporal and spatial information to reduce the number of search points. To utilize temporal information, some algorithms[32–34] perform MV prediction based on the motion vectors of the neighboring blocks in the previous frame. By doing so, the number of search points to be visited can be reduced.

On the other hand, some algorithms[32–35] use spatial information to speed up the MV estimation. Statistically, the MV of a block is highly correlated with the MVs of the neighboring blocks in the current frame. This may be explained by the assumption that objects usually span through several blocks and hence the MVs of the blocks will not differ too much when translational motion is considered. Therefore, only a portion of the search points need to be visited, and hence the process can be sped up.

In these algorithms, MVs of adjacent or neighboring blocks in the current frame or previous frame are used in the MV determination. However, they suffer from one major problem: they always select the MVs of the neighboring blocks in the current or previous frame as candidate MVs of the current block. This assumes that the MV of the current block must have relationship with the MVs of the neighboring blocks, which is not always the case. Some algorithms, like PSA,[35] predict the MV from the weighted average of the MVs of the neighboring blocks. This inherently assumes that the relationship between the MVs is a weighted average. In reality, these assumptions may not be true. The MV of a block may not be equal to one of the MVs of neighboring blocks or to the weighted average of the MVs of neighboring blocks.

In our research, we focus on the issues concerning the MV prediction based on the spatial information. Temporal information is important, but we believe that spatial information should be considered first.

3 Proposed Fast Motion Estimation

3.1 *Initial Considerations*

In a typical video sequence, there are many objects (including the background as an object) that span a group of blocks. Thus, the MVs of the blocks within this group must have some relationship between them. If the relationship between the MVs of the group of blocks can be identified, it is then possible to predict the target MV from the MVs of the group of blocks to which the target block belongs. To determine this relationship, object extraction may help, but it usually involves edge detection or segmentation, which makes it a computationally complex and time-consuming process. On the other hand, the MVs of the blocks can be considered to see if they belong to the same object. Furthermore, if the predicted MV is accurate enough, we only need to search for the MV candidates that do not differ much from the predicted MV, and hence the search space can be reduced.

3.2 *Assumptions*

As in block-based motion estimation, we assume that:

1. The motion in a typical video sequence consists of mainly translational and rotational motion.
2. The motion vector of a block represents the overall motion of the block.

In a typical video sequence, there are motions other than translational and rotational motion. The first assumption says that these kinds of motion can be modeled by block-based motion estimation. The second assumption says that the MV can well approximate most of the motion of the pixels within a block. Both assumptions are considered reasonable in a block-based sense.

3.3 *Approach Overview*

To find the MV, one first predicts it with the MV prediction model. The prediction model tests whether neighboring blocks lie on the same object, and if so, the MV is predicted from the MVs of the neighboring blocks. There can be more than one predicted MV; in this case the MV that gives the minimum MAE will be selected. The final predicted MV will be used to address a search center in the search window. After that, a search-center-biased search pattern will be used to locate the best match. An MV refinement technique will then be employed to find the final MV.[36] Figure 1 depicts the conceptual flow of the approach.

3.4 *Model for Motion Vector Prediction*

3.4.1 *Relationship of motion vectors of the blocks lying on the same object*

Consider two blocks A and B at two distinct respective locations A and B initially $(t=0)$, as depicted in Fig. 2. Suppose at time t, block A and block B are estimated to have moved to A' and B', respectively.

If block A and block B lie on the same object,

$$\|\overrightarrow{AB}\| = \|\overrightarrow{A'B'}\|$$

$$\Leftrightarrow \quad \|\overrightarrow{AB}\|^2 = \|\overrightarrow{A'B'}\|^2$$

$$\Leftrightarrow \quad X_{AB}^2 + Y_{AB}^2 = \|\overrightarrow{A'A} + \overrightarrow{AB} + \overrightarrow{BB'}\|^2$$

$$\Leftrightarrow \quad X_{AB}^2 + Y_{AB}^2 = \|-a_x(t)\mathbf{i} - a_y(t)\mathbf{j} + X_{AB}\mathbf{i} + Y_{AB}\mathbf{j} + b_x(t)\mathbf{i} + b_y(t)\mathbf{j}\|^2$$

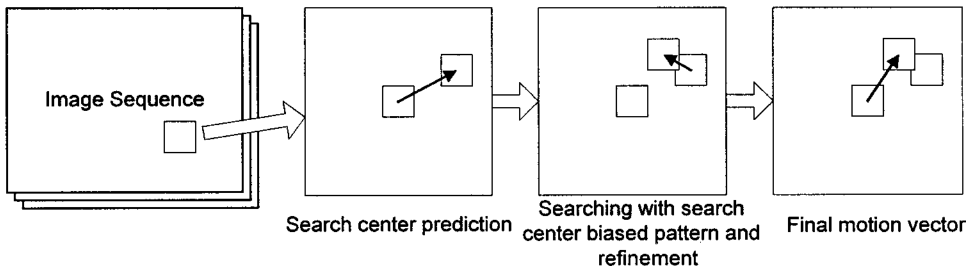


Fig. 1 Conceptual diagram

$$\Leftrightarrow \quad X_{AB}^2+Y_{AB}^2=[b_x(t)-a_x(t)+X_{AB}]^2 +[b_y(t)-a_y(t)+Y_{AB}]^2.$$

The above equation is true for all positive values of *t*. However, we are only interested in small values of *t*, because the frame time is usually small.

Call the right-hand side $R(t)$. Then, for small $t=\Delta t$,

$$R(\Delta t)=R(0)+\dot{R}(0)\ \Delta t+\text{h.o.t.}\approx X_{AB}^2+Y_{AB}^2+\dot{R}(0)\ \Delta t$$

$$\because\ \dot{R}(t)=2[b_x(t)-a_x(t)+X_{AB}][\dot{b}_x(t)-\dot{a}_x(t)] +2[b_y(t)-a_y(t)+Y_{AB}][\dot{b}_y(t)-\dot{a}_y(t)].$$

$$\therefore\ \dot{R}(0)=2X_{AB}[\dot{b}_x(0)-\dot{a}_x(0)]+2Y_{AB}[\dot{b}_y(0)-\dot{a}_y(0)].$$

$$\therefore\ R(\Delta t)=X_{AB}^2+Y_{AB}^2+2\{X_{AB}[\dot{b}_x(0)-\dot{a}_x(0)] +Y_{AB}[\dot{b}_y(0)-\dot{a}_y(0)]\}\ \Delta t.$$

Equating $R(t)$ with the left-hand side, we have

$$X_{AB}[\dot{b}_x(0)-\dot{a}_x(0)]+Y_{AB}[\dot{b}_y(0)-\dot{a}_y(0)]=0. \tag{6}$$

If the frame time Δt is small enough,

$$\dot{a}_x(0)\approx\frac{a_x(\Delta t)-a_x(0)}{\Delta t},\quad \dot{a}_y(0)\approx\frac{a_y(\Delta t)-a_y(0)}{\Delta t}, \tag{7}$$

$$\dot{b}_x(0)\approx\frac{b_x(\Delta t)-b_x(0)}{\Delta t},\quad \dot{b}_y(0)\approx\frac{b_y(\Delta t)-b_y(0)}{\Delta t}. \tag{8}$$

Substituting Eqs. (7), (8) into Eq. (6) yields

$$X_{AB}[b_x(\Delta t)-a_x(\Delta t)]+Y_{AB}[b_y(\Delta t)-a_y(\Delta t)]=0. \tag{9}$$

For small Δt, any two blocks that satisfy the above condition lie on the same object. Therefore, Eq. (9) can be used to test whether two blocks lie on the same object, and to predict the MV of the target block. Equations (7) and (8) actually attempt to approximate the velocities of the blocks by the their respective average velocities. For a typical video sequence, the frame time Δt is about 1/25 to 1/30 s, which is usually small enough.

3.4.2 *Prediction of target motion vector*

Consider a target block *T* and the neighboring blocks, *A*, *B*, *C*, *D* as depicted in Fig. 3. Let the

MV of *A* be	$a_x(\Delta t)\mathbf{i}+a_y(\Delta t)\mathbf{j}$
MV of *B* be	$b_x(\Delta t)\mathbf{i}+b_y(\Delta t)\mathbf{j}$
MV of *C* be	$c_x(\Delta t)\mathbf{i}+c_y(\Delta t)\mathbf{j}$
MV of *D* be	$d_x(\Delta t)\mathbf{i}+d_y(\Delta t)\mathbf{j}$
MV of *T* be	$t_x(\Delta t)\mathbf{i}+t_y(\Delta t)\mathbf{j}$.

If *T* lies on the same object as *A*, *B*, *C*, or *D*, the MV of *T* can be predicted from the MVs of A, B, C, or D. Although *T* may be predicted from blocks other than A, B, C, and D, which are not necessarily neighboring blocks, it is more likely that *T* and its neighboring blocks lie on the same object.

Consider four neighboring blocks. There are altogether 10 possible combinations of *T* lying on the same object together with one or a pair of the four neighboring blocks. There are also cases where *T* lies on the same object as more than two neighboring blocks. However, these cases can always be decomposed into two or more of the above cases. We first consider the case when *T* lies on the same object with one and only one of its neighboring blocks. Figure 4 explains each case and describes how the prediction of the MV is done in each.

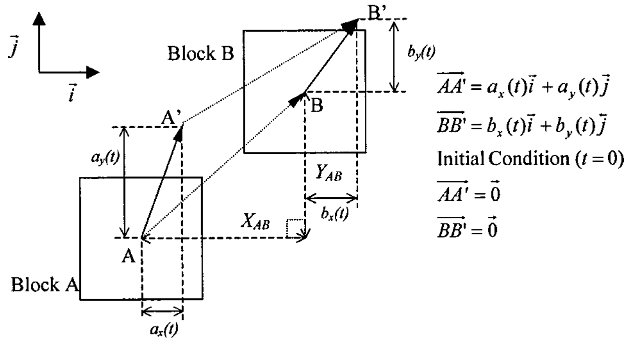

$\vec{j}$
$\vec{i}$
Block B
B'
$b_y(t)$
$\overrightarrow{AA'}=a_x(t)\vec{i}+a_y(t)\vec{j}$
$\overrightarrow{BB'}=b_x(t)\vec{i}+b_y(t)\vec{j}$
Initial Condition ($t=0$)
$\overrightarrow{AA'}=\vec{0}$
$\overrightarrow{BB'}=\vec{0}$
A'
B
$a_y(t)$
Y_{AB}
X_{AB}
$b_x(t)$
A
Block A
$a_x(t)$

Fig. 2 Prediction model

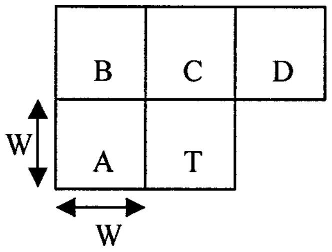


Fig. 3 Target block (T) and neighboring blocks (A, B, C, and D).

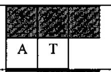
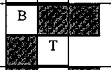
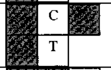
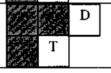
Case	Blocks Pair	MV Prediction
1		$t_x(\Delta t) = a_x(\Delta t)$ **(10)**
2		$t_x(\Delta t) - t_y(\Delta t) = b_x(\Delta t) - b_y(\Delta t)$ **(11)**
3		$t_y(\Delta t) = c_y(\Delta t)$ **(12)**
4		$t_x(\Delta t) + t_y(\Delta t) = d_x(\Delta t) + d_y(\Delta t)$ **(13)**

Fig. 4 MV prediction when *T* lies on the same object with one and only one neighboring block.

In each of the above cases, the MV of *T*, $(t_x(\Delta t), t_y(\Delta t))$, lies on a straight line defined by the corresponding equation in Fig. 4. Then, if *T* lies on the same object as one and only one of the block pair (A, B), (A, C), (A, D), (B, C), (B, D), and (C, D), the MV of *T* will be the intersection of the lines defined in the corresponding cases.

This idea can be extended to other cases as well. Figure 5 summarizes the cases where *T* lies on the same object as two neighboring blocks.

When *T* lies on the same object as three of the neighboring blocks, it can be decomposed into cases 5 to 10 in Fig. 5. For example, if *T* lies on the same object as blocks *A*, *B*, *C*, this can be decomposed into cases 5, 6, and 8, as depicted in Fig. 6.

When none of cases 5 to 10 holds (that is, there does not exist a pair of neighboring blocks that lie on the same object), we have no way to explicitly predict the MV of *T*, and

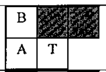
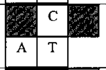
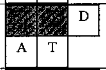
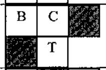
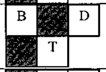
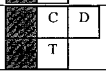
Case	Blocks Pair	MV Prediction
5		$(t_x(\Delta t), t_y(\Delta t)) = (a_x(\Delta t), b_y(\Delta t) - b_x(\Delta t) + a_x(\Delta t))$ **(14)**
6		$(t_x(\Delta t), t_y(\Delta t)) = (a_x(\Delta t), c_y(\Delta t))$ **(15)**
7		$(t_x(\Delta t), t_y(\Delta t)) = (a_x(\Delta t), d_x(\Delta t) + d_y(\Delta t) - a_x(\Delta t))$ **(16)**
8		$(t_x(\Delta t), t_y(\Delta t)) = (b_x(\Delta t) - b_y(\Delta t) + c_y(\Delta t), c_y(\Delta t))$ **(17)**
9		$(t_x(\Delta t), t_y(\Delta t)) = ((b_x(\Delta t) + d_x(\Delta t) + d_y(\Delta t) - b_y(\Delta t))/2, (b_y(\Delta t) + d_y(\Delta t) + d_x(\Delta t) - b_x(\Delta t))/2)$ **(18)**
10		$(t_x(\Delta t), t_y(\Delta t)) = (d_x(\Delta t) + d_y(\Delta t) - c_y(\Delta t), c_y(\Delta t))$ **(19)**

Fig. 5 MV prediction when *T* lies on the same object with one and only one of the neighboring block pair.

in that case the predicted MVs are considered to be the MVs of the neighboring blocks and (0,0). Figure 7 depicts one possibility where this happens.

3.4.3 *Further considerations*

First, although it is assumed that the two blocks lie on the same object whenever the MVs of two blocks satisfy Eq. (9) (because it is not a conformal mapping), even if the MVs satisfy Eq. (9), there is a small possibility that corresponding blocks may not lie on the same object.

Second, it is required to know which block lies on the same object as the target block. We may use Eq. (9) to do the testing for the different pairs of blocks involved. For

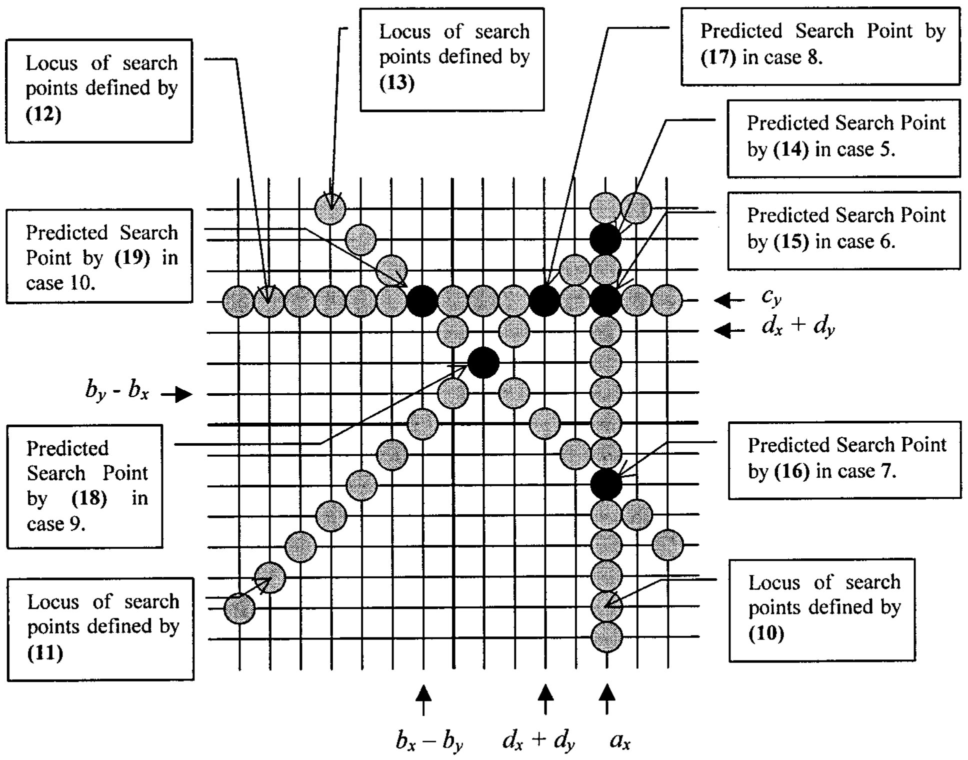

Locus of search points defined by (12)
Locus of search points defined by (13)
Predicted Search Point by (17) in case 8.
Predicted Search Point by (14) in case 5.
Predicted Search Point by (15) in case 6.
Predicted Search Point by (19) in case 10.
c_y
$d_x + d_y$
$b_y - b_x$
Predicted Search Point by (18) in case 9.
Predicted Search Point by (16) in case 7.
Locus of search points defined by (11)
Locus of search points defined by (10)
$b_x - b_y$
$d_x + d_y$
a_x

Fig. 6 Prediction of Search Points when T lies on the same object as more than two neighboring blocks.

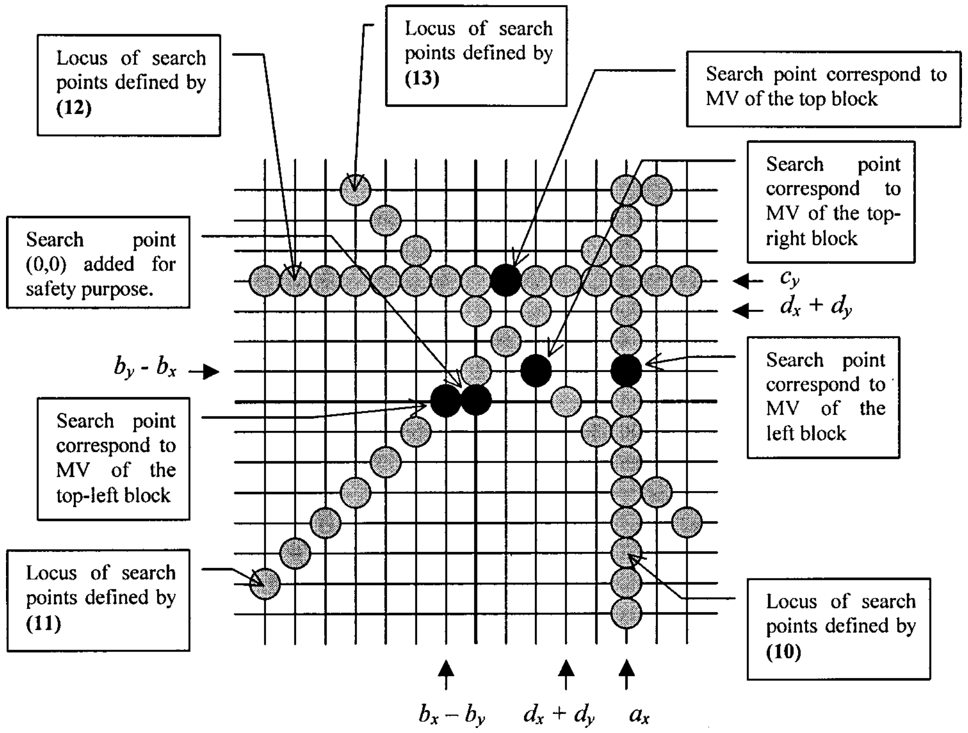

Locus of search points defined by (12)
Locus of search points defined by (13)
Search point correspond to MV of the top block
Search point correspond to MV of the top-right block
Search point (0,0) added for safety purpose.
c_y
$d_x + d_y$
$b_y - b_x$
Search point correspond to MV of the left block
Search point correspond to MV of the top-left block
Locus of search points defined by (11)
Locus of search points defined by (10)
$b_x - b_y$
$d_x + d_y$
a_x

Fig. 7 Search points when there exists no pair of adjacent blocks lie on the same object.

example, if we need to know whether block A, B, or C lies on the same object, we may use Eq. (9) to test whether any of the block pairs (A,B), (B,C), and (C,A) lie on the same object. But since the test is not a conformal mapping, we actually cannot conclude that A, B, or C lies on the same object even if all the three tests are positive. In this case, we can only conclude that it is likely that at least one block pair (A,B), (B,C), or (C,A) lies on the same object. Therefore, we should consider all the cases involved and take all the three MVs predicted in each case as possible MVs of T. The MV that gives the minimum MAE should be chosen as the predicted MV of T.

Third, for the testing the condition itself, the left-hand side of Eq. (9) does not necessarily equal zero even if the block pair involved really lies on the same object. Therefore, for practical applications, the testing conditions given in Eq. (9) should be modified as follows:

$$|X_{AB}[b_x(\Delta t)-a_x(\Delta t)]+Y_{AB}[b_y(\Delta t)-a_y(\Delta t)]| \leq \epsilon(X_{AB},Y_{AB}), \quad (10)$$

where $\epsilon(\cdot,\cdot)$ is the error tolerance function, which is always the upper bound of the error of the right-hand side of Eq. (9), and depends on X_{AB} and Y_{AB}.

Let Δe be the error of left-hand side of Eq. (9). Then

$$|\Delta e| \leq |X_{AB}|[|\Delta b_x(\Delta t)|+|\Delta a_x(\Delta t)|] + |Y_{AB}|[|\Delta b_y(\Delta t)|+|\Delta a_y(\Delta t)|].$$

For full-pixel precision, $|\Delta b_x(\Delta t)| \leq 0.5$, $|\Delta a_x(\Delta t)| \leq 0.5$, $|\Delta b_y(\Delta t)| \leq 0.5$, $|\Delta a_y(\Delta t)| \leq 0.5$, we have

$$|\Delta e| \leq |X_{AB}|(0.5+0.5)+|Y_{AB}|(0.5+0.5)=|X_{AB}|+|Y_{AB}|.$$

Therefore, we have to make $\epsilon(X_{AB},Y_{AB})=|X_{AB}|+|Y_{AB}|$ to ensure it is an upper bound of the error Δe. For half-pixel precision, $|\Delta b_x(\Delta t)| \leq 0.25$, $|\Delta a_x(\Delta t)| \leq 0.25$, $|\Delta b_y(\Delta t)| \leq 0.25$, $|\Delta a_y(\Delta t)| \leq 0.25$, we have

$$|\Delta e| \leq |X_{AB}|(0.25+0.25)+|Y_{AB}|(0.25+0.25) = 0.5(|X_{AB}|+|Y_{AB}|).$$

Therefore, we have to make $\epsilon(X_{AB},Y_{AB})=0.5(|X_{AB}|+|Y_{AB}|)$ to ensure it is an upper bound of error Δe.

Finally, we do not know whether T lies on the same object as one of its neighboring blocks, since the MV of T is unknown. In a typical video sequence, there is a large chance that a block lies on the same object as one of its neighboring blocks. Therefore, the proposed algorithm always assumes that T lies on the same object when we know that at least one pair of the neighboring blocks lie on the same object.

3.5 *Search-Point Pattern*

The distribution of MVs in a typical video sequence is highly biased towards the central region[20,21] of the search window. Thus, it is reasonable to place more search points in the central region of the search window to obtain more samples. However, that is true only when video sequences all consists of gentle motion. This may not be the case when there is a lot of fast motion or panning motion in the sequence. On the other hand, with accurate MV prediction, the distribution of actual MVs should be highly biased to-

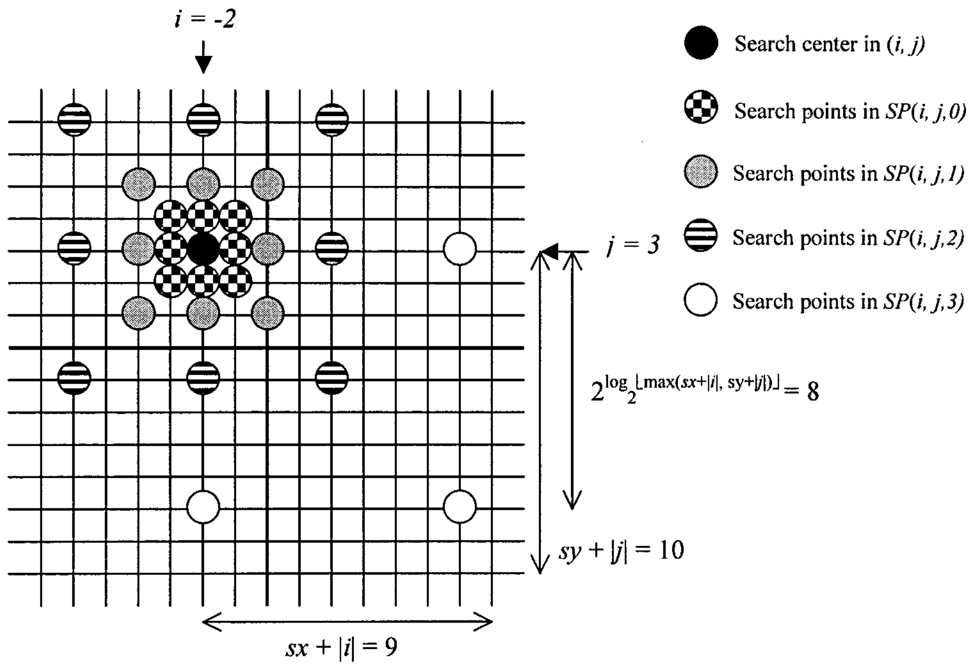

i = -2
Search center in (i, j)
Search points in SP(i, j, 0)
Search points in SP(i, j, 1)
Search points in SP(i, j, 2)
Search points in SP(i, j, 3)
j = 3
$2^{\log_2 \lfloor \max(sx+|i|,\ sy+|j|) \rfloor} = 8$
sy + |j| = 10
sx + |i| = 9

Fig. 8 Search Center Biased Search Pattern with search center $(-2,3)$, $sx=7$, and $sy=7$.

wards the predicted MV even without restricting the sequences to mainly gentle and mild motion. It is therefore reasonable to place more search points near the predicted search center, which is the search-point position in the search window addressed by the predicted MV.

3.5.1 *General search pattern*

The general search-pattern set at a search-point location (i,j) is defined as follows:

$$\mathrm{SP}(i,j,n)=\{(x,y):x=i+2^n p,\ \ y=j+2^n q$$

where

$$p,q\in\{-1,0,1\},\ |p|+|q|\neq 0,\quad \text{and}$$
$$-s_x\leq x\leq s_x\,,-s_y\leq y\leq s_y\},\tag{11}$$

where s_x and s_y are the half width and half height of the search window.

The general search patterns with different parameters can be combined to form more complex search patterns. With the general search pattern, we can proceed to define the search-center-biased search pattern.

3.5.2 *Search-center-biased search pattern*

The search-center-biased pattern around a search center (i,j) is defined as

$$\mathrm{CBSP}(i,j)=\{(i,j)\}\cup_{0\leq n\leq \log_2}\cup_{\lfloor \max(s_x+|i|,s_y+|j|)\rfloor}\mathrm{SP}(i,j,n)\tag{12}$$

The search-center-biased pattern is the union of general search patterns of different step sizes around the search center (i,j) plus the search center itself. In the definition of $\mathrm{CBSP}(i,j)$, the expression $\log_2\lfloor\max(s_x+|i|,s_y+|j|)\rfloor$ determines the maximum possible value of n, so that 2^n is the maximum possible step size. Figure 8 shows the search-center-biased search pattern when $s_x=s_y=7$ with search center $(i,j)=(-2,3)$.

As shown in the example, the density of search points is high in the region close to the search center and decreases with increasing distance from the search center.

The searching begins at the search center and then proceeds to $\mathrm{SP}(i,j,n)$ with increasing n. This makes an outward spiral search, and whenever the minimum is found within the spiral, the search will terminate. Figure 9 depicts the outward spiral search pattern used in our implementation. After searching with the search-center-biased search pattern, a position with minimum MAE is located. This gives a preliminary MV, from which MV refinement is required to find the final MV. The refinement process is the same as that in TSS. But the refinement will take a smaller initial step size when the minimum position is closer to the search center.

3.6 *MV Refinement*

After the search with $\mathrm{CBSP}(i_c,i_c)$, the preliminary MV has been obtained. It is only a coarse MV, and hence MV refinement is needed. Since the search pattern in the first step search is a center-biased one, the preliminary MV that corresponds to search points closer to the search center is less coarse than those that correspond to search points further away from the search center. Thus, the step size will be smaller if the minimum search-point position is closer to the predicted search center. The MV refinement process is actually the same as that in TSS. Figure 10 shows one possible refinement with search center at $(-2,1)$.

If the preliminary MV found in the first step lies at one of the corner points of the central 3×3 region around the search center (i_c,j_c), two more search points are visited.

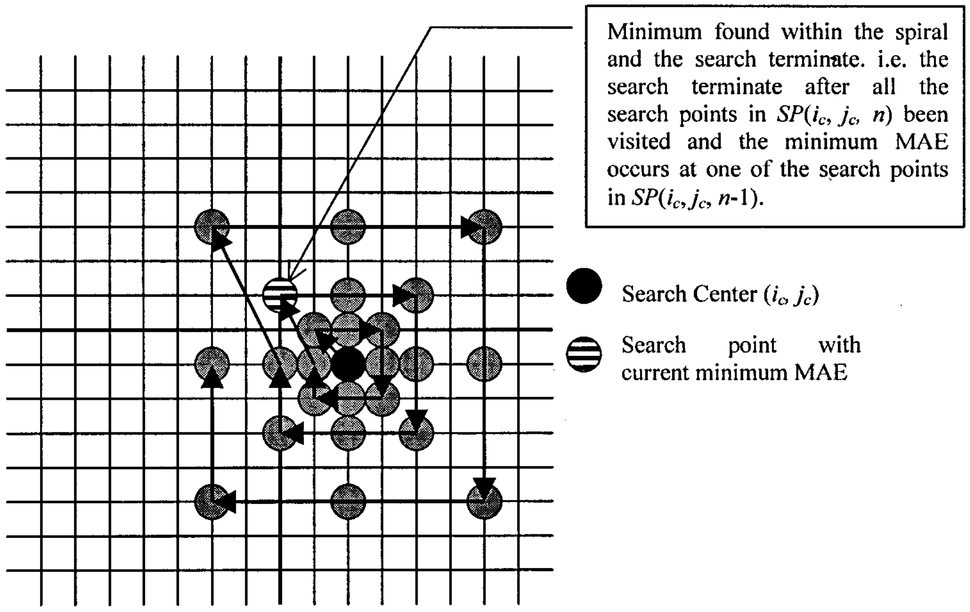

Minimum found within the spiral and the search terminate. i.e. the search terminate after all the search points in $SP(i_c, j_c, n)$ been visited and the minimum MAE occurs at one of the search points in $SP(i_c, j_c, n\text{-}1)$.
● Search Center (i_c, j_c)
Search point with current minimum MAE

Fig. 9 Outward Spiral Search Pattern with a search center (2,−1).

That is because at corner points there are only two directly neighboring search points, compared with four at the others. Therefore, we cannot guarantee that a corner point has minimum MAE among all its direct neighbors. The two additional search points are the corresponding directly neighboring search points. Figure 11 shows an example of this.

After this MV refinement, the final MV is obtained.

4 Results and Analysis

The algorithm was implemented and has been tested on four 90-frame MPEG test sequences: ''Football,'' ''Susie,'' ''Flower Garden,'' and ''Mobile and Calendar.'' The image size of ''Football'' and ''Susie'' is 720×480, whereas that of ''Flower Garden'' and ''Mobile and Calendar'' is 352×240. All the sequences are encoded into MPEG2 bitstreams using the MPEG2 encoder from MSSG[37] with the motion estimation algorithm changed to the proposed algorithm. Each group of pictures (GOP) in ''Football'' and ''Susie'' contains 15 frames, while those in ''Flower Garden'' and ''Mobile and Calendar'' contain 12 frames. The block size is 16×16, and 2 two-frame interpolation structure was used. The search range is −31 to 31 pixels for ''Football'' and ''Susie,'' and −15 to 15 for ''Flower Gar-

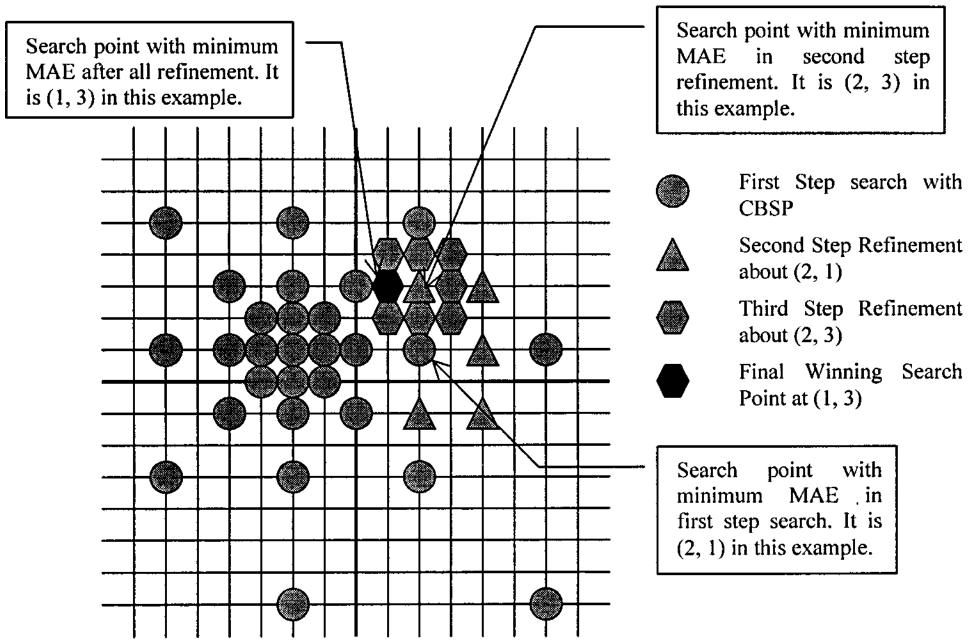

Search point with minimum MAE after all refinement. It is (1, 3) in this example.
Search point with minimum MAE in second step refinement. It is (2, 3) in this example.
First Step search with CBSP
Second Step Refinement about (2, 1)
Third Step Refinement about (2, 3)
Final Winning Search Point at (1, 3)
Search point with minimum MAE in first step search. It is (2, 1) in this example.

Fig. 10 Possible refinement of predicted MV with search center (−2,1).

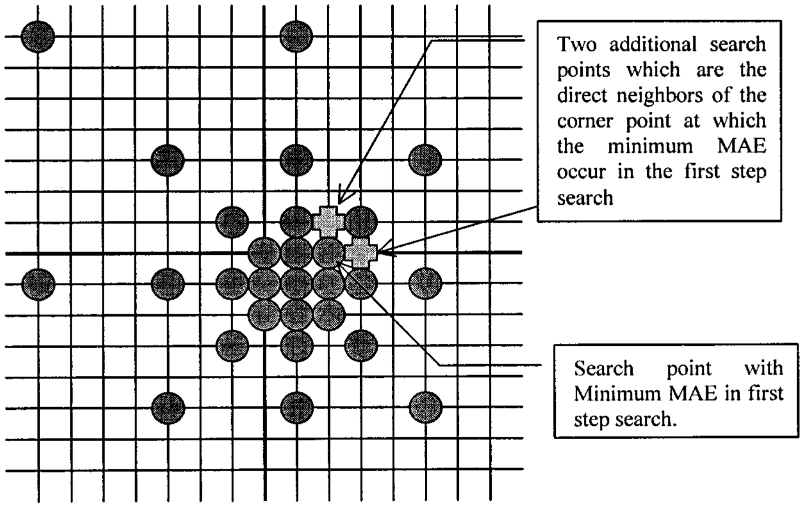

Two additional search points which are the direct neighbors of the corner point at which the minimum MAE occur in the first step search
Search point with Minimum MAE in first step search.

Fig. 11 Possible search pattern if the preliminary MV determined in the first step lies on the corner points in the central 3×3 region.

den'' and ''Mobile and Calendar.'' Other fast search algorithms such as TSS, NTSS, FSS, CS, 1DFS, and DS have also been implemented for comparing their performance with the algorithm proposed in this paper. The performance is evaluated on four counts: average PSNR, average MSE, the total number of search points (TNSP) visited, and the total time taken in the process of motion estimation (TME) for the whole sequence. The total number of search points is the sum of all the search points visited by the algorithm, both in forward and in backward prediction. The TMEs were measured on a Pentium II 333 machine running Linux in single-user mode. The TME of each algorithm shown in Table 1 is the average of five experiments.

As shown in Table 1 for the ''Flower Garden'' sequence, the proposed algorithm can achieve an average PSNR very close to that of FS. It achieves 99.77% of the latter PSNR. Similar results have been obtained in the other sequences, the proposed algorithm achieving 99.77% to 99.94% of the PSNR of FS. On the other hand, FSS and 1DFS can only achieve 96.36% to 99.79% and 98.52% to 99.79% of the PSNR of FS, respectively.

The proposed algorithm only requires 1.40% and 1.34% of the TNSP of FS for the sequences ''Football'' and ''Susie,'' respectively, and 4.07% and 3.54% for the sequences ''Flower Garden'' and ''Mobile and Calendar.'' Compared with CS, the proposed algorithm uses a 9.14% to 128.67% larger TNSP. Although the proposed algorithm is not as fast as CS in general for the sequences tested, it achieves much better PSNR.

As shown in the above, most algorithms have been successful in optimizing one objective measure like average PSNR, average MSE, TNSP, or TME. However, it is not fair to use just one objective measure to rank the algorithms. It is necessary to establish a metric to evaluate the performance of each algorithm. We proposed to normalize the average PSNR and TME of each algorithm to those of FS and plot the normalized average PSNR (NPSNR) versus the normalized TME (NTME). We plot the NTME instead of the NTNSP because the NTNSP does not include the overhead of search-center prediction and search-pattern generation. The results are given in Table 2.

Figure 12 depicts the performance of each algorithm for the ''Flower Garden'' sequence. In this diagram, the best algorithm would reside in the top left corner, and the worst in the bottom right corner. On the other hand, algorithms that optimize for quality by sacrificing speed would lie in the top right corner, whereas those that optimize for speed by sacrificing quality would lie in the bottom left corner.

As shown, CS is the fastest algorithm. However, it has the poorest PSNR among all the algorithms evaluated. On

Table 1 Average PSNR, average MSE, and TNSP for the test sequence ''Flower Garden.''

Algorithm	PSNR	MSE	TNSP	TME (μs)
FS	26.319	155.656	16,337,910	95,325,491
Proposed	26.259	157.930	664,860	4,932,502
TSS	25.250	198.210	1,218,592	8,629,149
NTSS	25.180	200.969	1,095,426	7,823,985
FSS	25.361	193.336	909,991	6,641,623
1DFS	26.240	158.744	2,441,837	16,182,176
CS	22.666	355.600	609,177	4,104,747
DS	25.297	196.142	824,566	5,818,942

Table 2 Normalized average PSNR, TNSP, and TME for ''Flower Garden.''

Algorithm	NPSNR	NTNSP	NTME
Proposed	0.9977	0.0407	0.0517
FS	1.0000	1.0000	1.0000
TSS	0.9594	0.0746	0.0905
NTSS	0.9567	0.0671	0.0821
FSS	0.9636	0.0557	0.0697
1DFS	0.9970	0.1495	0.1698
CS	0.8612	0.0373	0.0431
DS	0.9612	0.0505	0.0610

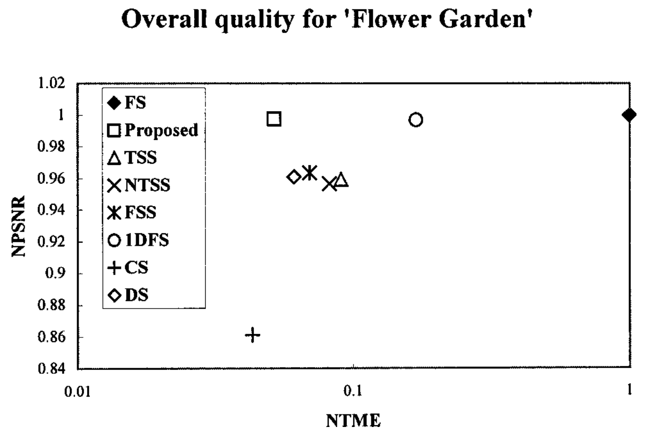


Fig. 12 Overall quality of each algorithm for the "Flower Garden" sequence.

the other hand, FS achieves the best PSNR, but is the slowest algorithm in the group. Thus algorithms that optimize either on speed or on quality do not come close to the top left corner. A good trade-off has to be made in order to place an algorithm close to the top left corner. Other algorithms like TSS, NTSS, FSS, and 1DFS lie closer to the top left corner, indicating that they have made better trade-offs. But among all the algorithms, the proposed algorithm is the closest to the top left corner. This shows that the proposed algorithm performs better than all the other algorithms, at least for this particular sequence.

The PSNR behavior of the proposed algorithm over the whole sequence is also important for evaluating its performance. To investigate this, the PSNR of the proposed algorithm, FS, and FSS are plotted versus the frame number. FS and FSS are chosen for comparison because FS is the common reference for PSNR comparison and FSS is the next best algorithm among the group of algorithms evaluated. Figure 13 depicts the PSNR of ''Flower Garden'' for the proposed algorithm, FS, and FSS.

Besides the objective measures, we are also interested in the subjective measure of the algorithms. Selected frames in the sequence ''Flower Garden'' are used to perform this subjective comparison. The proposed algorithm, FS, TSS, NTSS, FSS, CS, 1DFS, and DS are then applied to the selected frames to perform the motion estimation to extract the MVs first, and then reconstruct the frames from the

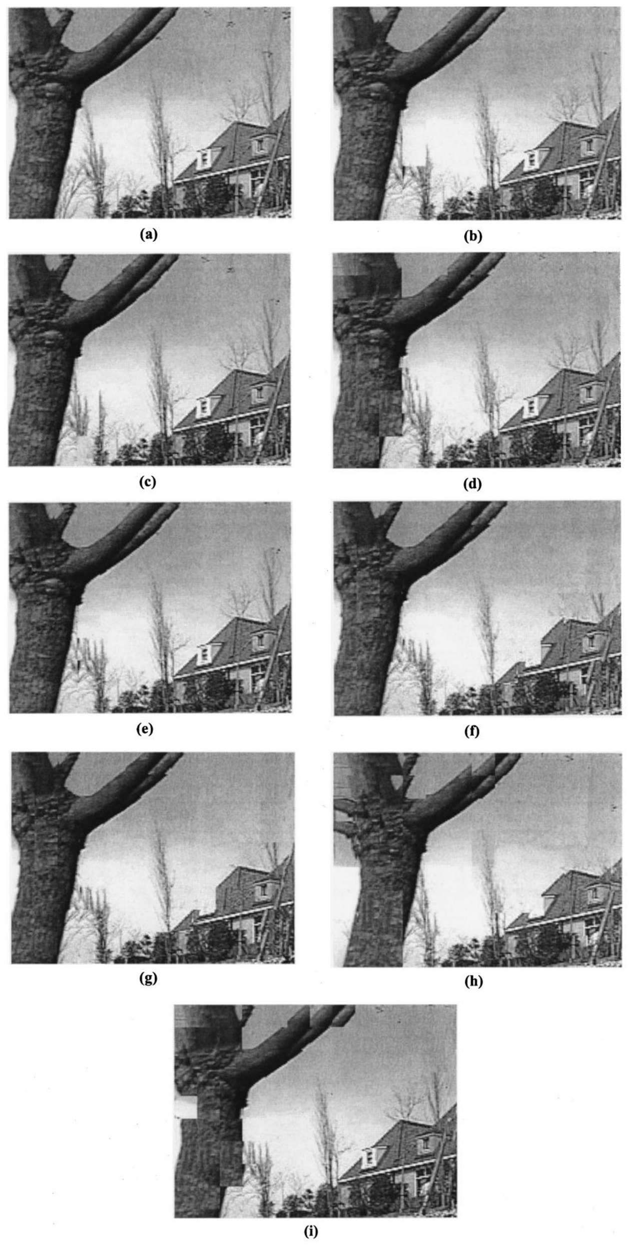


Fig. 14 Original and reconstructed frames by each algorithm: (a) original, (b) FS, (c) proposed, (d) FSS, (c) 1DFS, (f) NTSS, (g) TSS, (h) CS, (i) DS.

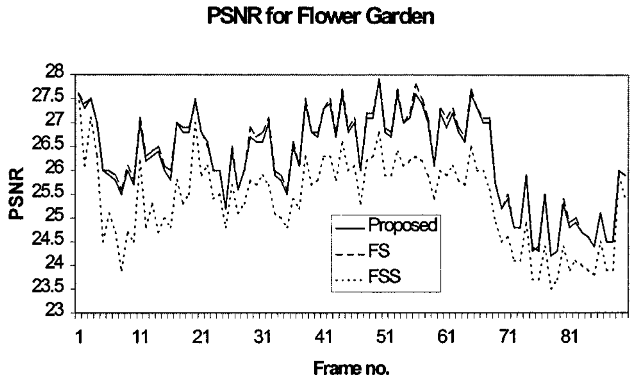


Fig. 13 PSNR behavior for the "Flower Garden" sequence.

MVs determined by each algorithm without error compensation. The reconstructed frames are then compared with each other. Figure 14 shows the frames in ''Flower Garden'' as reconstructed by all the algorithms.

From these reproductions, it can be seen that the frames reconstructed by FS, the proposed algorithm, and 1DFS resemble to the original frame most, although some of the details are missed. As for the other algorithms, they cannot preserve the details of the roof and the edges of the tree, showing that they can be easily trapped by a nonoptimum solution. Although 1DFS shows almost the same visual performance as the proposed algorithm, it is much slower. This shows that the proposed search-center prediction method can help in putting the initial search point sufficiently close to the optimum solution to reduce the chance of being trapped in a local minimum, and hence a lot less search points are required.

5 Conclusion and Future Directions

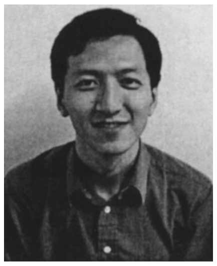
In this paper, a fast motion estimation algorithm based on search-center prediction and a search-center-biased pattern has been proposed. The algorithm predicts the search center by using a model, which formulates the relationship between the MVs of neighboring blocks lying on the same object, before the MV search process. It is found that the predicted search center is sufficient close to the optimum solution so that a search-center-biased search pattern can be used to speed up the MV searching process.

We have evaluated the proposed algorithm with four video sequences, and its performance has been compared with six other motion estimation algorithms. The results give strong support to the belief that our proposed algorithm offers a good trade-off between speed and quality. The proposed algorithm can achieve a PSNR very close to FS with a lot less search points. From the test results on the four sequences it is found that the proposed algorithm is able to achieve over 99.7% of the PSNR of FS while requiring less than 4.1% of the computation time. When compared with the other six algorithms, the proposed algorithm achieves the best PSNR performance. Although the proposed algorithm is not as fast as CS on three of the sequences, its PSNR performance is far better than that of CS.

Future research will be focused on improving both the prediction model and the search pattern so that even greater speedup and better PSNR performance can be achieved. We shall consider incorporating different kinds of motion such as zooming and the motions that originate from deformable objects, so that better MV prediction can be achieved. Also, the search pattern will be refined to further reduce the number of search points, which should lead to greater speedup.

References

1. K. Lengwehasatit and A. Ortega, ‘‘A novel computationally scalable algorithm for motion estimation,’’in *Proc. VCIP-97*, *Proc. SPIE* **3309**, 68–79 (1997).
2. B. Zeng, R. Li, and M. L. Liou, ‘‘Optimization of fast block motion estimation algorithms,’’ *IEEE Trans. Circuits Syst. Video Technol.* **7**(6), 833–844 (1997).
3. A. N. Netravali and J. O. Limb, ‘‘Picture coding: a review,’’ *Proc. IEEE* **PROC-68**, 366–406 (1980).
4. C.Cafforio and F. Rocca, ‘‘The differential method for motion estimation,’’ in *Image Sequence Processing and Dynamic Scene Analysis*, T. S. Huang, Ed., pp. 104–124, Springer-Verlag, New York (1983).
5. D. R. Walker and K. R. Rao, ‘‘Improved pel-recursive motion compensation,’’ *IEEE Trans. Commun.* **COM-32**, 1128–1134 (1984).
6. B. K. P. Horn and B. G. Schunck, ‘‘Determining optical flow,’’ *Artif. Intell.* **17**, 185–203 (1981).
7. B. Lucas and T. Kanade, ‘‘An iterative image registration technique with an application to stereo vision,’’ *DARPA Image Understanding Workshop*, pp. 121–130 (1981).
8. H.-H. Nagel, ‘‘Displacement vectors derived from second-order intensity variations in image sequences,’’ *Comput. Graph. Image Process.* **21**, 85–117 (1983).
9. D. J. Heeger, ‘‘Optical flow using spatio-temporal filters,’’ *Int. J. Comput. Vis.* **1**, 279–302 (1998).
10. D. J. Fleet and A. D. Jepson, ‘‘Computation of component image velocity from local phase information,’’ *Int. J. Comput. Vis.* **5**, 77–104 (1990).
11. CCITT SGXV, ‘‘Description of reference model 8 (RM8),’’ Document 525, Working Party XV/4, Specialists Group on Coding for Visual Telephony (1989).
12. ITU Telecommunication Standardization Sector LBC-95, Study Group 15, Working Party 15/1, Expert’s Group on Very Low Bitrate Visual Telephony, from Digital Video Coding Group, Telenor Research and Development (1995).
13. ISO/IEC CD 11172-2 (MPEG-1 Video), ‘‘Information technology—coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbits/s: video’’ (1993).
14. ISO/IEC CD 13818-2—ITU-T H.262 (MPEG-2 Video), ‘‘Information technology—generic coding of moving pictures and associated audio information: video’’ (1995).
15. F. Dufaux and F. Moscheni, ‘‘Motion estimation techniques for digital TV: a review and a new contribution,’’ *Proc. IEEE* **83**(6), 858–875 (1995).
16. T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, ‘‘Motion compensated interframe coding for video conferencing,’’ in *IEEE Proc. Nat. Telecommun. Conf.*, pp. G5.3.1–G5.3.4 (1981).
17. J. N. Kim and T. S. Choi, ‘‘A fast three-step search algorithm with minimum checking points using unimodal error surface assumption,’’ *IEEE Trans. Consum. Electron.* **44**(3), 638–648 (1998).
18. Y. H. Fok, O. C. Au, and R. D. Murch, ‘‘Novel fast block motion estimation in feature subspace,’’ *IEEE Proc. ICIP’95*, Vol. 2, pp. 209–212 (1995).
19. Y.-L. Chan and W.-C. Siu, ‘‘Edge oriented block motion estimation for video coding,’’*IEE Proc. Vision Image Signal Process.* **144**, 136–144 (1997).
20. A. C. K. Ng and B. Zeng, ‘‘A new fast motion estimation algorithm based on search window sub-sampling and object boundary pixel block matching,’’ *IEEE Proc. ICIP’98*, pp. 605–608 (1998).
21. B. Tao and M. T. Orchard, ‘‘Feature-accelerated block matching,’’ *Proc. SPIE* **3309**, 469–476 (1998).
22. S. Zhong, F. Chin, Y. S. Cheung, and D. Kwan, ‘‘Hierarchical motion estimation based on visual patterns for video coding,’’ *IEEE Proc. ICASSP’96*, pp. 2325–2328 (1996).
23. T. Koga, ‘‘Motion-compensated inter-frame coding for video conferencing,’’ in *IEEE Proc. NTC81*, pp. C9.6.1–C9.6.5 (1981).
24. M. Ghanbari, ‘‘The cross-search algorithm for motion estimation,’’ *IEEE Trans. Commun.* **38**(7), 950–953 (1990).
25. M. J. Chen, L. G. Chen, and T. D. Chiueh, ‘‘One-dimensional full search motion estimation algorithm for video coding,’’ *IEEE Trans. Circuits Syst. Video Technol.* **4**(5), 504–509 (1994).
26. R. Srinivasan and K. R. Rao, ‘‘Predictive coding based on efficient motion estimation,’’ in *IEEE Proc. ICC’84*, pp. 521–526 (1984).
27. R. Li, B. Zeng, and M. Liou, ‘‘A new three-step search algorithm for block motion estimation,’’ *IEEE Trans. Circuits Syst. Video Technol.* **4**(4), 438–442 (1994).
28. M. P. Lai and W. C. Ma, ‘‘A novel four-step search algorithm for fast block motion estimation,’’ *IEEE Trans. Circuits Syst. Video Technol.* **6**(3), 313–317 (1996).
29. S. Zhu and K. K. Ma, ‘‘A new diamond search algorithm for fast block-matching motion estimation,’’ *IEEE Trans. Image Process.* **9**(2), 287–290 (2000).
30. J. Y. Tham, S. Ranganath, M. Ranganath, and A. A. Kassim, ‘‘A novel unrestricted center-biased diamond search algorithm for block motion estimation,’’ *IEEE Trans. Circuits Syst. Video Technol.* **8**(4), 369–377 (1998).
31. J. Lu and M. L. Liou, ‘‘A simple and efficient search algorithm for block-matching motion estimation,’’ *IEEE Trans. Circuits Syst. Video Technol.* **7**(2), pp. 429–433 (1997).
32. S. Kim, J. Chalidabhongse, and C.-C. J. Kuo, ‘‘Fast motion estimation by using spatio-temporal correlation of motion field,’’ *Proc. SPIE* **2501**, 810–821 (1995).
33. J. Chalidabhongse, S. Kim, and C.-C. J. Kuo, ‘‘Fast motion estimation for video coding based on multiresolution-spatio-temporal correlations,’’ *Proc. SPIE* **2727**, 645–656 (1996).
34. J. Chalidabhongse and C.-C. J. Kuo, ‘‘Fast motion estimation using multiresolution-spatio-temporal correlations,’’ *IEEE Trans. Circuits Syst. Video Technol.* **7**(3), 477–488 (1997).
35. L. Luo, C. Zou, and X. Gao, ‘‘A new prediction search algorithm for block motion estimation in video coding,’’ *IEEE Trans. Consum. Electron.* **43**(1), 56–61 (1997).
36. H. Y. Chung, P. Y. S. Cheung, N. H. C. Yung, ‘‘Adaptive search center non-linear three step search,’’ *IEEE Proc. ICIP’98*, Vol. 2, pp. 191–194 (1998).
37. MPEG Software Simulation Group, MPEG2 Encoder v1.2.



Hing Yip Chung received his BEng in computer engineering and his MPhil degree from the Department of Electrical and Electronic Engineering, University of Hong Kong, in 1997 and 1999, respectively, where he is currently working toward the PhD degree. His research interests include digital image processing and video technology.

N. H. C. Yung received his BSc and PhD degrees from the University of Newcastle-Upon-Tyne while he was a Croucher Foundation Scholar in 1982 and 1985, respectively. He was a lecturer at the same university from 1985 until 1990, where he successfully completed a number of national and European research projects on image processing and robotics, either as leader or principle investigator. From 1990 to 1993, he worked as a senior research scientist at the Department of Defense, Australia, where he headed a team on the R&D of military-grade signal analysis systems. He joined the University of Hong Kong in late 1993 as an associate professor. He leads a research team in digital image processing and Intelligent Transportation Systems, and a number of research collaborative projects with various companies and organizations in the HKSAR, China, Japan and Australia. He is the founding director of the Laboratory for Intelligent Transportation Systems Research at HKU. Dr. Yung has co-authored a computer vision book, and has published over 100 research journal and conference papers in the areas of digital image processing, parallel algorithms, visual traffic surveillance and autonomous vehicle navigation. He serves as reviewer for the *IEEE Transactions of SMC, Signal Processing, IEE Pt. G, Optical Engineering, HKIE Proceedings, Mocroprocessors and Microsystems*, and *Robotics and Autonomous Systems Journal.* He is a member of the Advisory Panel of the *ITS Strategy Review*, Transport Department, HKSAR, and a Council Member of ITS-HK. He is a chartered electrical engineer, member of the HKIE and IEE and senior member of the IEEE. He was regional secretary of the IEEE Asia-Pacific Region from 1995 to 1997.

P. Y. S. Cheung received his BSc degree in engineering and his PhD degree from the Imperial College of Science and Technology, University of London, in 1973 and 1978, respectively. He worked for Queen's University of Belfast and the Hong Kong Polytechnic before joining the University of Hong Kong in 1980, where he served as Associated Dean of Engineering from 1991 to 1994 and Dean of the Faculty of Engineering from 1994 to 2000. He was elected as the Asia Pacific Region Director of IEEE in 1995-96 and was the first person from Hong Kong to serve as the Institute's Secretary of IEEE in 1997. Over the years he has graduated over 20 research students and published over 60 research papers. His research includes parallel computer architecture, Internet computing, VLSI design, signal processing and pattern recognition. He is a chartered engineer and a senior member of the IEEE. He was an independent non-executive director of Pacific Century CyberWorks from 1999 to 2000. He is also a director of Versitech, a technology transfer company of the University of Hong Kong, and was a founding director of the Information Technology Entrepreneur Association, a non-profit organization to promote I.T. entrepreneurs. He is also the honorary program director of the MSc in E-commerce and Internet Computing at the University of Hong Kong. Since October 2000, he has also been with Pacific Century CyberWorks as senior vice-president of technology.