Chapter 2
Performance Evaluation of Block Matching Algorithms for Video Coding

A video sequence typically contains temporal redundancy; that is, two successive pictures are often very similar except for changes induced by object movement, illumination, camera movement, and so on. Motion estimation and compensation are used to reduce this type of redundancy in moving pictures. The block-matching algorithm (BMA) for motion estimation has proved to be very efficient in terms of quality and bit rate; therefore, it has been adopted by many standard video encoders. In this chapter, the basic principle of block matching motion estimation and compensation is introduced and fast motion search algorithms are addressed.

2.1 Search Algorithms for Motion Estimation

There exist two basic approaches to motion estimation:

1. Pixel-based motion estimation;
2. Block-based motion estimation.

The pixel-based motion estimation approach seeks to determine motion vectors for every pixel in the image. This is also referred to as the 'optical flow method', which works on the fundamental assumption of brightness constancy, that is, the intensity of a pixel remains constant when it is displaced. However, no unique match for a pixel in the reference frame is found in the direction normal to the intensity gradient. It is for this reason that an additional constraint is also introduced in terms of the smoothness of velocity (or displacement) vectors in the neighborhood. The smoothness constraint makes the algorithm interactive and requires excessively large computation time, making it unsuitable for practical and real-time implementation.

An alternative and faster approach is the block-based motion estimation. In this method, the candidate frame is divided into nonoverlapping blocks (of size

S. Metkar and S. Talbar, *Motion Estimation Techniques for Digital Video Coding*, SpringerBriefs in Computational Intelligence,
DOI: 10.1007/978-81-322-1097-9_2, © The Author(s) 2013

16 × 16, or 8 × 8, or even 4 × 4 pixels in the recent standards) and for each such candidate block, the best motion vector is determined in the reference frame. Here, a single motion vector is computed for the entire block, whereby we make an inherent assumption that the entire block undergoes translational motion. This assumption is reasonably valid, except for the object boundaries. Block-based motion estimation is accepted in all the video coding standards proposed till date. It is easy to implement in hardware and real-time motion estimation and prediction is possible.

The effectiveness of compression techniques that use block-based motion compensation depends on the extent to which the following assumptions hold:

- The illumination is uniform along motion trajectories.
- The problems due to uncovered areas are neglected.

For the first assumption it neglects the problem of illumination change over time, which includes optical flow but does not correspond to any motion. The second assumption refers to the uncovered background problem. Basically, for the area of an uncovered background in the reference frame, no optical flow can be found in the reference frame. Although these assumptions do not always hold for all real-world video sequences, they continue to be used as the basis of many motion estimation techniques.

2.2 Principle of Block Matching Algorithm

The block matching technique is the most popular and practical motion estimation method in video coding. Figure 2.1 shows how the block matching motion estimation technique works. Each frame of size $M \times N$ is divided into square blocks $B\ (i, j)$ of size $(b \times b)$ with $i = 1\ldots., M/b$ and $j = 1\ldots..N/b$. For each block B_m in the current frame, a search is performed on the reference frame to find a matching based on a block distortion measure (BDM). The motion vector (MV) is the displacement from the current block to the best matched block in the reference frame. Usually, a search window is defined to confine the search. The same motion vector is assigned to all pixels within block.

$$\forall \vec{r} \in B(i,j), \quad \vec{d}(\vec{r}) = \vec{d}(i,j) \tag{2.1}$$

where the image intensity at pixel location $\vec{r} = (d_x, d_y)^T$ and at time t is denoted by $I(\vec{r}, t)$ and $\vec{d} = (d_x, d_y)^T$ is the displacement during the time interval Δt.

Suppose a block has size $b \times b$ pixels and the maximum allowable displacement of an MV is $\pm w$ pixels both in horizontal and vertical directions, there are $(2w + 1)^2$ possible candidate blocks inside the search window. The basic principle of block matching algorithm is shown in Figs. 2.2 and 2.3.

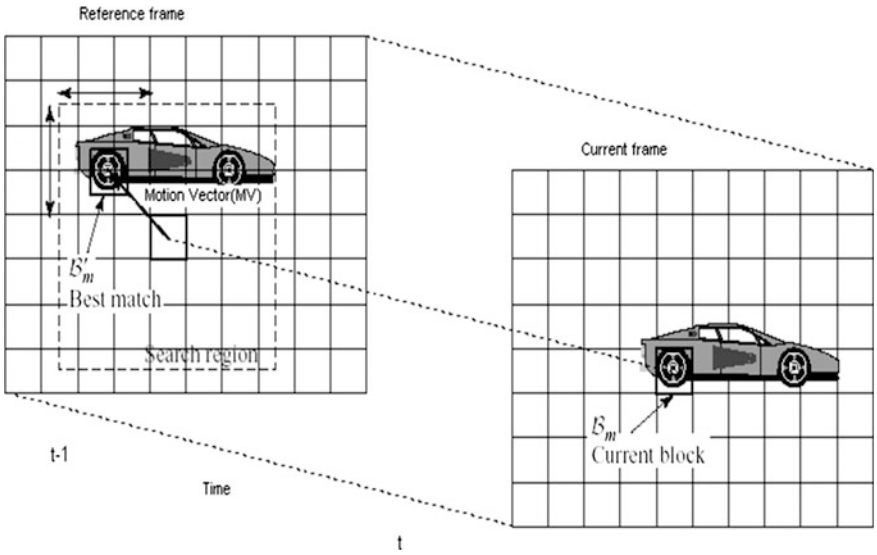

Fig. 2.1 Block matching motion estimation

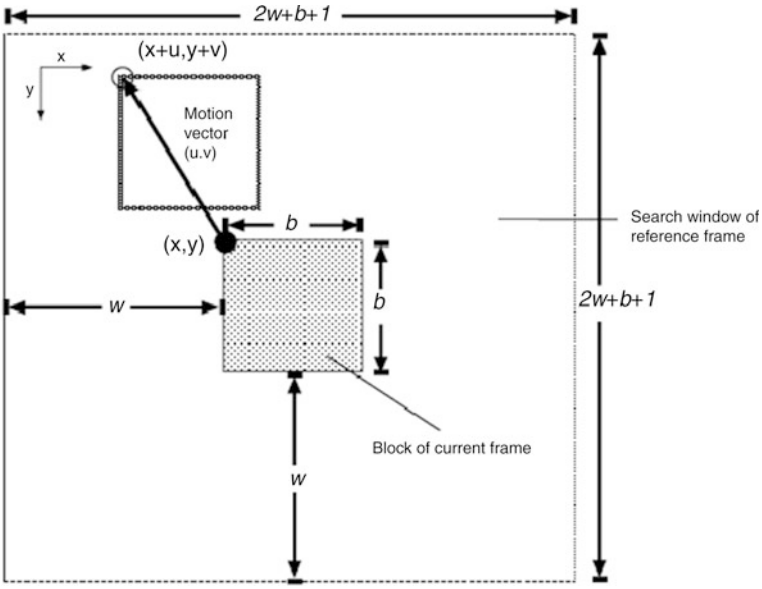

Fig. 2.2 Block matching method

A matching between the current block and one of the candidate blocks is referred to as a point being searched in the search window. If all the points in a search window are searched, the finding of a global minimum point is guaranteed.

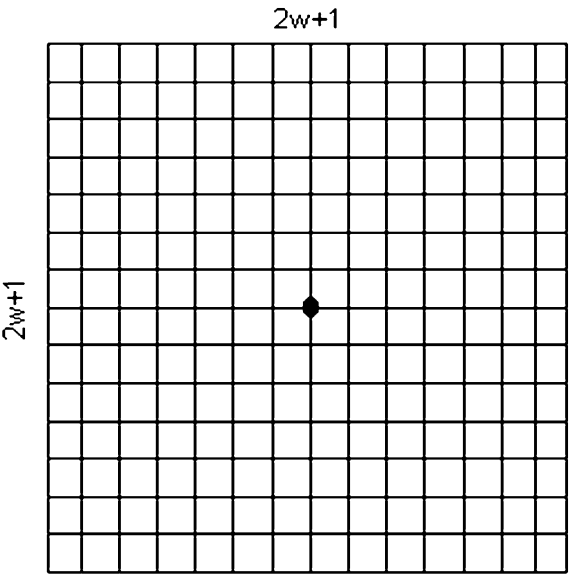

Fig. 2.3 Search point in a search window

There are different parameters of the BMA with impact on performance and accuracy in motion estimation and compensation. The first important parameter is the distortion function, the other is the block size, and finally, the maximum allowed motion displacement, also known the search range. All these parameters are elaborated in the following sections.

2.2.1 Block Distortion Measure

In order that the compressed frame looks like the original, the substitute block must be as similar as possible to the one it replaces. Thus, a matching criterion or distortion function is used to quantify the similarity between the target block and candidate blocks.

Assume, F_t is the current frame and F_{t-1} is the reference frame. $F(x, y)$ is the intensity of a pixel at (x, y) in frame F. Each candidate block is located at $\left(x + w_x, y + w_y\right)$ inside a search window of size $\pm w$ pixels such that $-w \leq w_x, wy \leq +w$. The optimum motion vector which minimizes BDM function is (u, v). There are a number of criteria to evaluate the "goodness" of a match. Popular matching criteria used for block-based motion estimation are:

- Mean Square Error (MSE);
- Sum of absolute difference (SAD).

- **Mean Square Error (MSE)**
  The mean square error of a block of pixels computed at a displacement (w_x, w_y) in the reference frame is given by

$$MSE(w_x, w_y) = \frac{1}{N \times N} \sum_{i=x}^{x+N-1} \sum_{j=y}^{y+N-1} \left[F(i,j) - F_{t-1}(i + w_x, j + w_y)\right]^2 \tag{2.2}$$

The MSE is computed for each displacement position (w_x, w_y) within a specified search range in the reference frame, and the displacement that gives the minimum value of MSE is the displacement vector which is more commonly known as motion vector and is given by

$$(u, v) = \min_{-w \le w_x, w_y \le +w} MSE(w_x, w_y) \tag{2.3}$$

The MSE criterion defined in Eq. (2.2) requires computation of N^2 subtractions, N^2 multiplications (squaring), and $(N^2 - 1)$ additions for each candidate block at each search position. MSE is the Euclidian distance between current and reference blocks. It is considered to be better BDM because it is closer to our visual perception. The drawback of MSE is that it is more complex than other distortion measures as it needs square operations.

- **Sum of Absolute Difference (SAD)**
  Similar to the MSE criterion, the sum of absolute difference (SAD) too makes the error values as positive, but instead of summing up the squared differences, the absolute differences are summed up. The SAD measure at displacement (w_x, w_y) is defined as

$$SAD(w_x, w_y) = \sum_{i=x}^{x+N-1} \sum_{j=y}^{y+N-1} \left|F(i,j) - F_{t-1}(i + w_x, i + w_y)\right| \tag{2.4}$$

The motion vector is determined in a manner similar to that for MSE as

$$(u, v) = \min_{-w \le w_x, w_y \le +w} SAD(w_x, w_y) \tag{2.5}$$

The SAD criterion shown in Eq. (2.4) requires N^2 computations of subtractions with absolute values and N^2 additions for each candidate block at each search position. The absence of multiplications makes this criterion computationally more attractive for real-time implementation.

2.2.2 Block Size

Another important parameter of the block matching technique is the block size. If the block size is smaller, it achieves better prediction quality. This is due to a number of reasons. A smaller block size reduces the effect of the accuracy problem. In other words, with a smaller block size, there is less possibility that the block will contain different objects moving in different directions. In addition, a smaller block size provides a better piecewise translational approximation to non-translational motion. Since a smaller block size means that there are more blocks (and consequently more motion vectors) per frame, this improved prediction quality comes at the expense of a larger motion overhead. Most video coding standards use a block size of 16×16 as a compromise between prediction quality and motion overhead.

2.2.3 Search Range

The maximum allowed motion displacement 'w' also known as the search range, has a direct impact on both the computational complexity and the prediction quality of the block matching technique. A small 'w' results in poor compensation for fast-moving areas and consequently poor prediction quality. A large '$\pm w$' on the other hand, results in better prediction quality but leads to an increase in the computational complexity (since there are $(2w + 1)^2$ possible blocks to be matched in the search window). A larger 'w' can also result in longer motion vectors and consequently a slight increase in motion overhead [1]. In general, a maximum allowed displacement of $w = \pm 7$ pixels is sufficient for low-bit-rate applications. MPEG standard uses a maximum displacement of about ± 15 pixels, although this range can optionally be doubled with the unrestricted motion vector mode.

2.3 Full Search Algorithm

One of the first algorithms to be used for block-based motion estimation is full search algorithm (FSA), which examines exhaustively all positions in the search area. The FSA is optimal in the sense that if the search range is correctly defined, it is guaranteed to determine the best matching position. However, if the search range in either direction is 'w' with step size of 1 pixel and assumes the search range is square, there are in total $(2w + 1)^2$ times of displacement in order to find a motion vector for each block, requiring a large amount of computations, especially for a large search window. The high computational requirements of FSA make it

unacceptable for real-time software-implemented video applications. For real-time implementation, quick and efficient search strategies were explored.

2.4 Fast Block Matching Algorithms

Many fast search algorithms have been proposed to reduce the computational complexity of FSA while retaining similar prediction quality. All of them make use of the quadrant monotonic model [2]. The quadrant monotonic model, first used for block matching by Jain and Jain, assumes that the value of the distortion function increases as the distance from the point of minimum distortion increases. Therefore, not only the candidate blocks close to the optimal block better match than those far from it, but also the value of the distortion function is a function of the distance from the optimal position. Thus, the quadrant monotonic assumption is a special case of the principle of locality. The quadrant monotonic assumption allows for the development of suboptimal algorithms that examine only some of the candidate blocks in the search area. In addition, they use the values of the distortion function to guide the search toward a good match. As the entire candidate blocks are not examined, the match found might not be the best available. But the trade-off between the quality of the match and the number of matching criteria evaluations is usually good. Some of the popular fast block matching algorithms are discussed in the following sections.

2.4.1 Two-Dimensional Logarithmic Search Algorithm

The two-dimensional logarithmic search (TDL), introduced by Jain and Jain in 1981, was the first block-matching algorithm to exploit the quadrant monotonic model to match blocks [3]. The initial step size 's' is $\lceil w/4 \rceil$ (where $\lceil\ \rceil$ is the upper integer truncation function) where 'w' is the search range in either direction. The block at the center of the search area and the four candidate blocks at a distance 's' from the center on the x and y axes are compared to the target block to determine the best match. The five positions form a pattern similar to the five points of a Greek cross (+). Thus, if the center of the search area is at position [0, 0], then the candidate blocks at position $[0, 0], [0, +s], [0, -s], [-s, 0]$, and $[+s, 0]$ are examined. Figure 2.4 shows the search pattern of 2D-logarithmic search algorithm.

The step size is reduced by half only when the minimum distortion measure point of the previous step is the center (cx, cy) or the current minimum point reaches the search window boundary. Otherwise, the step size remains the same. When the step size is reduced to one, all eight blocks around the center position

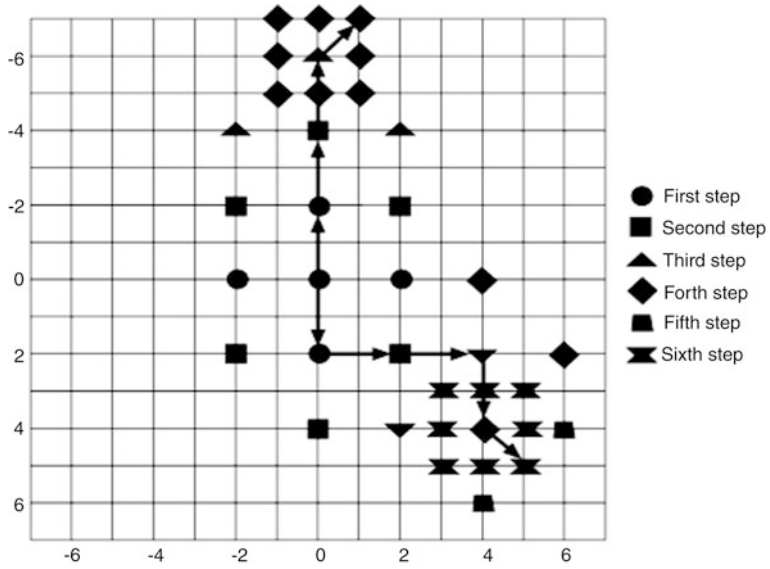

Fig. 2.4 2D-logarithmic search

which are $[cx - 1, cy - 1]$, $[cx - 1, cy]$, $[cx - 1, cy + 1]$, $[cx, cy - 1]$, $[cx, cy]$, $[cx, cy + 1]$, $[cx + 1, cy - 1]$, $[cx + 1, cy]$, and $[cx + 1, cy + 1]$ are examined, minimum distortion measure point of these is determined to be the best match for the target block and then it halts the algorithm. Otherwise (step size greater than one), the candidate blocks at positions $[cx, cy]$, $[cx + s, cy]$, $[cx - s, cy]$, $[cx, cy + s]$, and $[cx, cy - s]$ are evaluated for distortion measure. An experimental result proves that the algorithm performs well in large motion sequences because search points are quite evenly distributed over the search window.

2.4.2 Three-Step Search Algorithm

This algorithm is based on a coarse-to-fine approach with logarithmic decreasing in step size as shown in Fig. 2.5. The three-step search algorithm (TSS) tests eight points around the center [4].

For a center $[cx, cy]$ and step size 'd' the positions $[cx - d, cy - d]$, $[cx - d, cy]$, $[cx - d, cy + d]$, $[cx, cy - d]$, $[cx, cy]$, $[cx, cy + d]$, $[cx + d, cy - d]$, $[cx + d, cy]$, $[cx + d, cy + d]$ are examined. After each stage, the step size is halved and minimum distortion of that stage is chosen as the starting center of the next stage. The procedure continues till the step size becomes one. In this manner, TSS reduces the number of searching points as equal to $[1 + 8\{\log_2 (d + 1)\}]$. One problem that occurs with the TSS is that it uses a uniformly allocated checking point pattern in the first step, which becomes inefficient for small motion estimation.

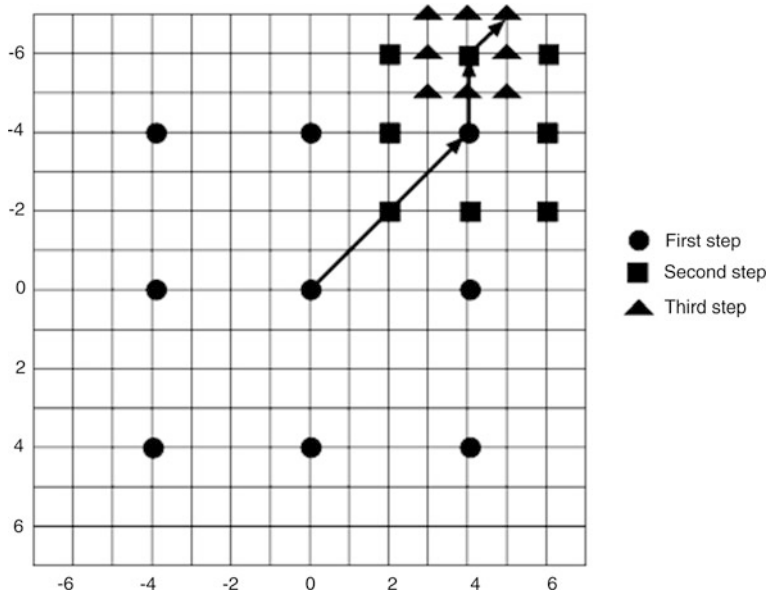

Fig. 2.5 Three-step search algorithm

2.4.3 Cross Search Algorithm

The cross search algorithm (CSA) proposed by Ghanbari [5] is a logarithmic step search algorithm using a saltire cross (×) searching patterns in each step. The CSA is presented in Fig. 2.6. The initial step size is half of maximum motion displacement '*w*'. At each stage, the step size is halved, until the final stage is equal to one. At the final stage, however, the end points of a Greek cross (+) are used to search areas centered around the top-right and bottom-left corners of the previous stage, and a saltire cross (x) is used to search areas centered around the top-left and bottom-right corners of the previous stage.

The CSA requires $[5 + 4\lceil \log_2 w \rceil]$ comparisons where '*w*' is the largest allowed displacement. The algorithm has a low computational complexity. It is, however, not the best in terms of motion compensation.

2.4.4 One-at-a-Time Search Algorithm

The One-at-a-time Search Algorithm (OTA) is a simple but effective algorithm, which has a horizontal and a vertical stage [6]. OTA starts its search at search window center. The center points and its two horizontally adjacent points, i.e., (0, 0) (0, −1) and (0, +1) are searched. If the smallest distortion is for the center points, start the vertical stage, otherwise look at the next point in the horizontal direction

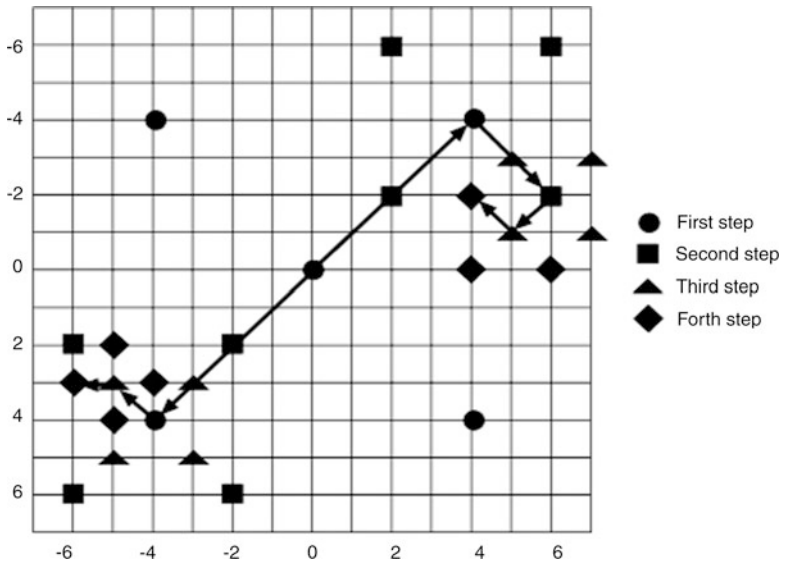

Fig. 2.6 Cross search algorithm

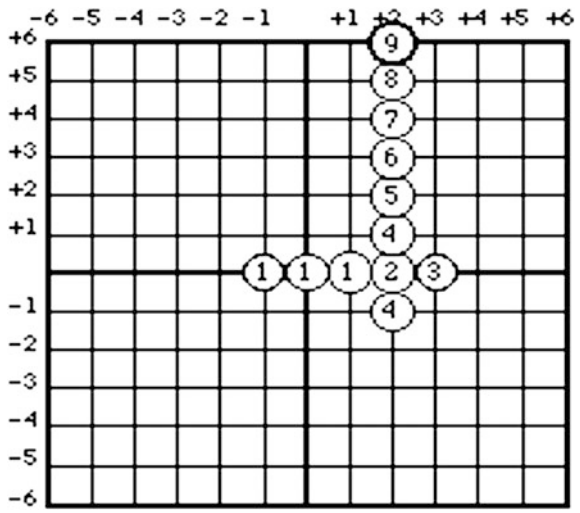

Fig. 2.7 One-at-a-time algorithm

closer to the point with the smallest distortion, and continue in that direction till you find the point with the smallest distortion. The step size is always one. OTA stops when the minimum distortion point is closeted between two points with higher distortion. The above procedure is repeated in vertical direction about the point that has the smallest distortion in the horizontal direction. The search pattern of OTA is shown in Figure 2.7. This search algorithm requires less time, however, the quality of the match is not very good.

Performance evaluation of FSA, TDL, 3SS, CSA, and OTA algorithms in terms of quality and computational complexity is discussed in Sect. 2.7.

2.5 Proposed Modified Algorithms

2.5.1 New One-at-a-Time Algorithm

In this section, a modified version of OTA, called modified OTA, is proposed. It outperforms OTA in terms of computational complexity as compared to OTA algorithm. As compared to OTA, instead of evaluating in horizontal direction till the minimum distortion point is closeted between two points with higher distortion, the proposed algorithm checks four points around the center in the horizontal direction. Initial checking points are $(i, j-1)$ $(i, j-2)$ $(i, j+1)$ and $(i, j+2)$ around the center (i, j). The step size is always one. If optimum is found at the center, further procedure stops pointing motion vector as (i, j). This will save more than 80 % of the computational time. Otherwise, we proceed to search around the point in vertical direction where the minimum was found. Figure 2.8 illustrates the NOTA to find the positions of minimum distortion.

The steps of the proposed NOTA are explained as follows:

Step-I Evaluate the objective function for all five points in the horizontal direction.

Step-II If the minimum occurs at the center, stop the search; the motion vector points to the center.

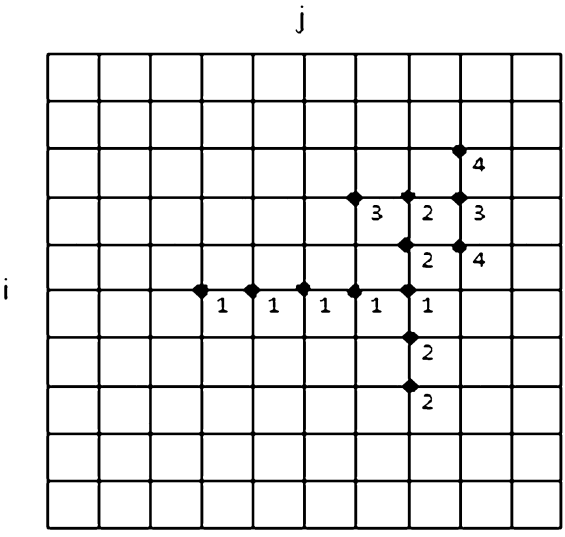

Fig. 2.8 New one-at-a-time algorithm

Step-III Otherwise, evaluate the objective function at four points on either side of previous (having minimum distortion of Step I winning point in the vertical direction around the winning point.).

Step-IV Search the two positions on either side of the winning point of step III horizontally; as regards the point that has the smallest distortion in the horizontal direction search the two positions vertically.

Step-V Minimum distortion point is declared as the best match.

The proposed NOTA retains the simplicity and regularity of OTA. Although NOTA uses more checking points in the first step as compared to OTA, pre-emption of the algorithm at the first step reduces the computations significantly. The experimental results of the proposed NOTA as compared to OTA are reported in Sect. 2.7.

2.5.2 Modified Three-Step Search Algorithm

The three-step search algorithm has been widely used in the block matching motion estimation due to its simplicity and effectiveness. However a TSS uses uniformly allocated checking point pattern in its first step, which is inefficient for the estimation of small motions, hence a modified three-step search algorithm is proposed. The features of MTSS are that it employs a center-biased checking point pattern in the first step, and a halfway-stop technique to reduce the computational cost.

The search procedure for MTSS is shown in Fig. 2.9. We have considered the search window size as ± 7.

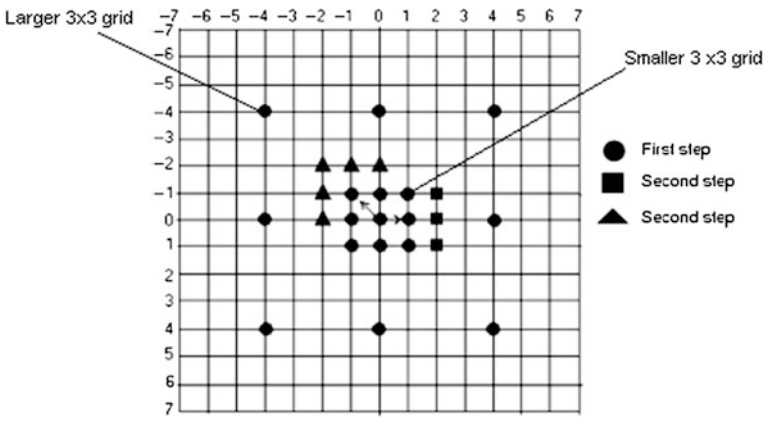

Fig. 2.9 Two different search paths of MTSS

The MTSS algorithm utilizes a smaller 3 × 3 grid search pattern at the center in addition to the larger 3 × 3 grid in the TSS. Thus distortion is evaluated for 17 search points. If the minimum BDM point is found at the center of the search window, the search will stop.

If the minimum BDM point is one of the eight points on the smaller 3 × 3 grid, only additional five or three points will be checked which depends on previous minimum distortion point. Otherwise, the search window center will be moved to the winning point on the larger 3 × 3 grid and the remaining procedure is same as in TSS. The detail of the algorithm is explained in the following steps.

Step-I For the first Step of MTSS along with larger 3 × 3 grid search points as in TSS, additional eight search points on smaller 3 × 3 grid at center, with step size equals to one are evaluated. This way, total 8 + 9 = 17 search points needs to be evaluated in the first Step. If the minimum BDM point is the search window center, the search will be terminated; otherwise algorithm proceeds for Step II.

Step-II If one of the central eight neighboring points on the 3 × 3 grid is found to be the minimum in the first Step, go to Step III; otherwise go to Step IV.

Step-III Move the smaller 3 × 3 grid so that the window center is the winning point found in Step I. Evaluate additional five or three points according to the location of the previous winning point, and then the search is stopped declaring minimum BDM point as the winning point.

Step-IV Reduce the step size of larger 3 × 3 grid by half and move the center to the minimum BDM point in Step I, hence forward, procedure of TSS algorithm is followed till step size becomes one.

Figure 2.9 shows two different search paths for finding motion vector within 7 × 7 search area. According to the halfway-stop technique, the M3SS needs to evaluate 17 search points for stationary blocks and 20 or 22 points for small motion within central 3 × 3 search area. For the worst case 25 + 8 = 33 points will be required, compared to 25 points in TSS. It has been experimentally proved that proposed MTSS shows good performance as compared to 3SS for slow motion sequence. The results of MTSS are discussed in Sect. 2.7.

2.6 Video Sequences for Simulation

In this research work, ten standard Quarter Common Intermediate File Format (QCIF) video sequences of different motion contents are used for performance comparison of different algorithms. These video sequences are categorized into three classes; Class A, Class B, and Class C, with increasing motion complexity.

Silent Claire Grandma

Fig. 2.10 First frames of video sequences "silent", "claire", and "grandma"

News Suzie Miss America

Fig. 2.11 First frames of video sequences "news", "suzie", and "miss" America

That is, the video sequences in Class A have low or slow motion activities. Those in Class B have medium motion activities and Class C videos have high or complex motion activities. The video sequences of Silent, Claire, and Grandma are all of slow object translation with low motion activities and belong to Class A. Their first frames are shown in Fig. 2.10. The first frames of the Class B sequences, News, Suzie, and Miss America with moderate motion are revealed in Fig. 2.11. Similarly, Fig. 2.12 presents the first frames of Foreman, Carphone, Salesman, and Trevor sequence having fast object translation with high motion activity which belongs to Class C.

2.7 Experimental Results

In this section, the performance of FSA, 3SS, 2-D logarithmic search, and OTA is discussed along with the proposed algorithm from the viewpoint of prediction of accuracy as well as the computational complexity.

Fig. 2.12 First frames of video sequences foreman, carphone, salesman, and trevor

2.7.1 Experimental Setup and Performance Evaluation Criterion

All the algorithms have been tested on desktop computer P-IV 2.4 GHz CPU. In our simulation block distortion measure (BDM) is defined to be the Mean Square Error (MSE). The block size is considered as 8×8 as tradeoffs between computational complexity and quality. The maximum motion in rows and column is assumed to be ± 7. Analysis has been done using three standard video sequences in QCIF (176×144) format, each representing different class of motion. These include Silent, News, and Foreman. The first 100 frames of the above-mentioned sequences have been used for simulation.

The quality of the reconstructed sequence should be estimated by subjective tests. One of the subjective metrics is Mean Square Error (MSE) which is evaluated between original frame and reconstructed frame. The lesser the value of MSE, the better the prediction quality. Mean Square Error is given by

$$MSE(i,j) = \frac{1}{M \times N} \sum_{m=1}^{M} \sum_{n=1}^{N} \left(f(m,n) - f'(m,n)\right)^2 \tag{2.6}$$

where $f(m,n)$ represents the current frame and $f'(m,n)$ is the reconstructed frame with frame size as M × N. Another widely used metric for comparing various image compression techniques is the peak-signal-to-noise-ratio (PSNR). The mathematical formulae for PSNR is

$$PSNR = 10\log_{10}(\frac{(2^b-1)^2}{MSE}) \tag{2.7}$$

The b in the equation is the number of bits in a pixel. For 8-bit uniformly quantized video sequence $b = 8$. The higher the value of PSNR, the better the quality of the compensated image.

Within the same search window, FSA can achieve the highest prediction quality because it searches all possible search points in the search window and thus is guaranteed to find optimum global minimum point. Therefore, prediction measure achieved by FSA is often used as reference to compare or evaluate other fast block matching algorithms. We consider the computational complexity of the algorithm in terms of the CPU time required to terminate the evaluation of the search algorithm. The time required increases linearly with the number of points in the search window being searched.

2.7.2 Performance Comparisons of Fast Block Matching Algorithms

In this section, we present experimental results to evaluate the performance of the FSA, TDLS, CSA, and OTA along with the proposed algorithms from the viewpoint of computational complexity as well as prediction accuracy.

From Table 2.1 it is found that for all sequences FSA shows higher PSNR values as compared to other algorithms with increased CPU time. MTSS has quality improvement in terms of PSNR over 3SS except for Foreman sequence. For example, the PSNR of MTSS is 37.99 and 37.34 db, higher than that of 3SS for sequences Silent and News, respectively. Coarse searches in TDLS and 3SS can locate the rough position of the global minimum and subsequent four searches can find the best motion vector. They perform well in large motion sequence because search points are evenly distributed over the search window. For higher motion, Foreman sequence the PSNR for TDLS is 33.44 and 33.27 db for 3SS which is higher than other algorithms apart from FSA. However, TDLS required higher CPU time for computation as compared to other fast algorithms. OTA performs one-dimensional gradient descending search on the error surface twice. Although it desires less computational time as compared with other fast block matching algorithms, its prediction quality is low which is reflected in the PSNR entries. This is because one-dimensional gradient descend search is insufficient to provide a correct estimation of the global minimum position. With slight improvement in the speed as compared to OTA, the proposed NOTA achieves

Table 2.1 Performance of block matching algorithms

Algorithms	Avg. MSE	Avg. PSNR	Avg. CPU Time in Sec.
Using silent sequence			
FSA	12.54	37.86	15.70
3SS	14.84	37.31	1.15
OTA	27.57	36.29	0.37
CSA	18.44	36.41	0.48
TDLS	15.97	37.11	1.43
NOTA	23.02	36.63	0.33
MTSS	14.42	37.34	0.76
Using news sequence			
FSA	17.10	38.14	15.70
3SS	18.86	37.88	1.26
OTA	24.89	37.53	0.27
CSA	22.88	37.59	0.44
TDLS	20.05	37.91	1.39
NOTA	22.95	37.47	0.22
MTSS	18.82	37.99	0.78
Using foreman sequence			
FSA	26.36	34.22	15.70
3SS	34.13	33.27	1.11
OTA	42.49	31.59	0.50
CSA	50.98	30.79	0.52
TDLS	32.06	33.44	1.58
NOTA	41.65	32.22	0.41
MTSS	38.14	33.20	0.92

higher PSNR quality for slow to fast motion sequence. The PSNR of OTA is 36.29 and 31.59 db for Silent and Foreman sequence and for NOTA, PSNR values are 36.63 and 32.22, respectively. In comparison with the other fast block matching algorithms, while the computational complexity of the CSA is the second lowest, its compensation performance is not.

Figure 2.13 shows comparisons of the MSE and PSNR per frame for the first 100 frames by using various search algorithms (including FSA, 3SS, TDLS, CSA, OTA, and proposed NOTA and MTSS) for Foreman, News, and Silent sequence.

2.8 Summary

In this chapter the importance of motion compensation in video coding is reviewed. To find motion vectors, block matching motion estimation (BMME) is performed. BMME takes up to 70 % for the total encoding time in modern video coding standards. The simplest algorithm for BMME is full search algorithm which can also achieve the best matching quality. However, the computational complexity

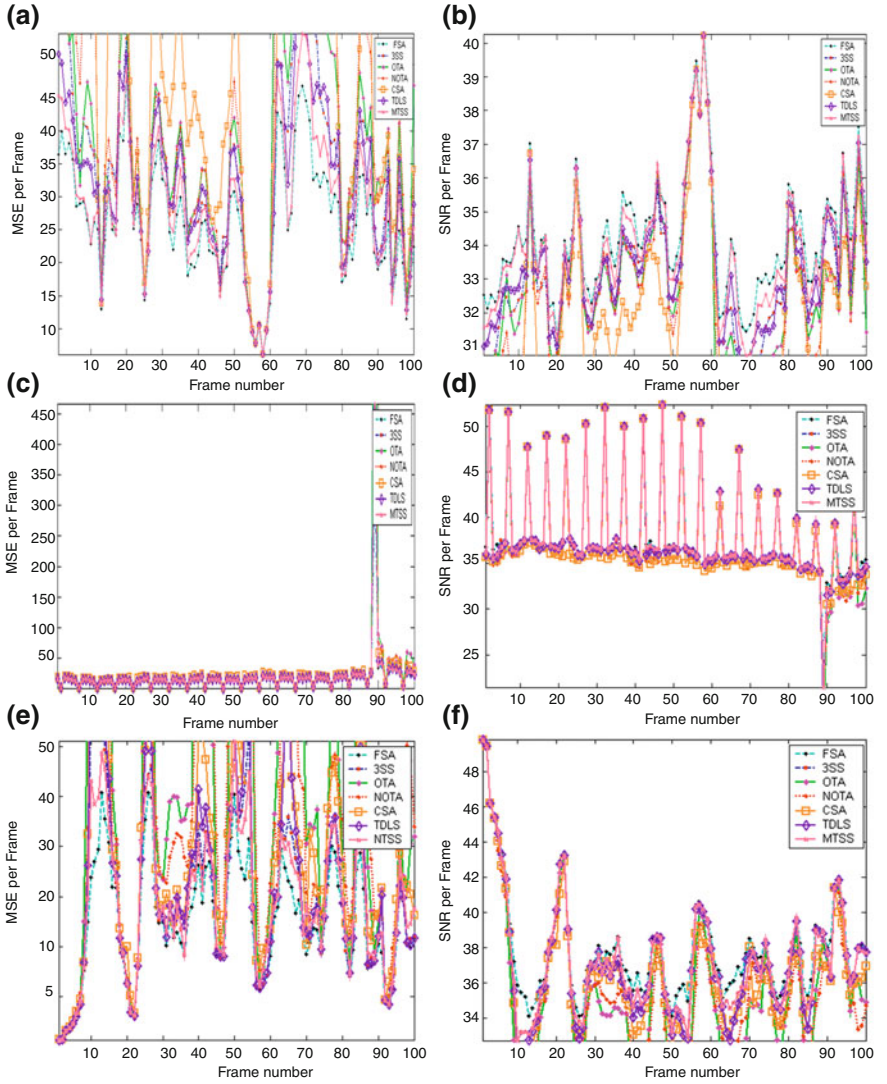

Fig. 2.13 Performance comparisons of different algorithms in terms of average MSE and average PSNR: (**a**) (**b**) Foreman (**c**) (**d**) News (**e**) (**f**) Silent

of full search algorithm is so high that it is unsuitable for many applications, e.g., real-time encoding. Fast block matching algorithms are proposed to reduce the computational complexity of FSA while maintaining similar matching quality. Some well-known algorithms which include 3SS, TDLS, CSA, and OTA are analyzed. To compare the performance of fast block matching algorithms, both matching quality and computational complexity have been considered. Matching quality is measured by Mean Square Error and Peak-Signal-to-Noise Ratio.

Computational complexity is measured by time required for motion estimation. Finally, video sequences used for analysis and simulation in this work are illustrated.

References

1. B. Liu, A. Zaccarin, New fast algorithms for the estimation of block motion vectors. IEEE Trans. Circuits Syst. Video Technol. **3**, 440–445 (1995)
2. International Organization for Standardization. *ISO/IEC 15938*-5:*2003*: *Information Technology—Multimedia Content Description Interface—Part 5*: *Multimedia Description Schemes*, 1st edn. Geneva, Switzerland, 2003
3. J.R. Jain, A.K. Jain, Displacement measurement and its application in interframe image coding. IEEE Trans. Commun. **29**(12), 1799–1808 (1981)
4. T. Koga, T. Ishiguro, Motion compensated inter-frame coding for video conferencing, *Proceedings of National Telecommunication Conference, New Orleans,* pp. G5.3.1–G5.3.5, Dec 1981
5. M. Ghanbari, The cross search algorithm for motion estimation. IEEE Trans. Commun. **38**(7), 950–953 (1990)
6. R. Srinivasan, K. R. Rao., Predictive coding based on efficient motion estimation. IEEE Trans. Commun. **33**(8), 888–896(1985)

http://www.springer.com/978-81-322-1096-2

Motion Estimation Techniques for Digital Video Coding

Metkar, S.; Talbar, S.

2013, XII, 64 p. 32 illus., Softcover
ISBN: 978-81-322-1096-2