

Chapter 2

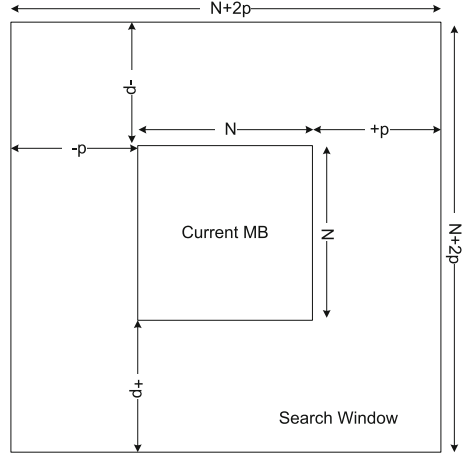
Background and Literature Survey

This chapter begins with an overview of block matching algorithm (BMA) approach to motion estimation which is preferred for its simplicity and straightforward circuit implementation. Many block matching algorithms are briefly introduced and also a brief survey of different motion estimation architectures are presented.

2.1 Block Matching Algorithm

Mainly, there are two different techniques of ME, namely Pel-Recursive Algorithm (PRA) and Block Matching Algorithm (BMA). In PRAs, there is an iterative refining of ME for individual pixels by gradient methods [1]. On the other hand, BMAs assume that all the pixels within a block have the same motion activity [2]. In BMAs, motion is estimated on the basis of rectangular blocks and one Motion Vector (MV) is produced for each block. Compared to BMAs, PRAs involve more computational complexity and less regularity, and so are difficult to realize in hardware. In general, BMAs are more suitable for a simple hardware realization because of their regularity and simplicity [3]. Also, BMA is adopted in all video coding standards because of its performance [4]. In the process of BMA, one is required to find a MB in the reference frame within a given search area, that is most similar to the MB in the current frame (current MB). Due to a given search range a window like structure is formed in the reference frame, which is known as the Search Window (SW). For a search range of $[-p, +p]$ and for a MB of size $N \times N$, the spatial relationship between the current MB and the SW is shown in Fig. 2.1. The matching criterion of the BMA has a direct impact on coding efficiency and computational complexity. Many matching criteria have been proposed in literature e.g., mean squared error, Sum of Absolute Differences (SAD), pel difference classification etc. [5]. Among the various proposed matching criteria, SAD calculation requires only a few simple computational steps, and thus is the most preferred one for VLSI

Fig. 2.1 The process of motion estimation by block matching algorithm



implementation. The evaluation of SAD for a given location (m, n) within the SW is done as:

$$SAD(m, n) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} |cur(i, j) - ref(i + m, j + n)| \quad (2.1)$$

where $-p \leq m, n \leq +p$. Also, $cur(i, j)$ is the current MB of size $N \times N$ at the coordinate location (i, j) , while $ref(i + m, j + n)$ is the reference block within the SW at the coordinate location $(i + m, j + n)$ and p is the search range in both the directions. The term $|cur(i, j) - ref(i + m, j + n)|$ is known as the distortion which is the absolute difference in intensity between the current pixel $cur(i, j)$ and the reference pixel $ref(i + m, j + n)$ [4]. The expression $SAD(m, n)$ yields the summation of all the distortions for the current MB at the search location (m, n) . The search candidate, which has the smallest SAD, is selected as the best matching reference MB, and the associated location (m, n) is the MV of this current MB. In the following subsections, ME algorithms are broadly classified into two categories, namely full search algorithm and fast search algorithms based on the provided quality and the search process.

2.1.1 Full Search Block Matching Algorithm

In full search BMA, all search candidates within the search window are evaluated, and the search candidate with the smallest SAD is selected as the best matched search candidate. The final MV is obtained from the location of the best matched search candidate. The algorithm for full search based ME is shown in Algorithm 2.1. Since all the search candidates are examined in full search, ME based on the full

search provides the optimum solution. Although full search yields optimum results, it requires a huge amount of computation.

Algorithm 2.1 Full Search Block Matching Algorithm

```

1:  $SAD_{min} = MAXVALUE$ ;
2:  $MV = (0, 0)$ ;
3: for  $m = -p$  to  $+p$  do
4:   for  $n = -p$  to  $+p$  do
5:      $SAD(m, n) = 0$ ;
6:     for  $i = 0$  to  $N-1$  do
7:       for  $j = 0$  to  $N-1$  do
8:          $SAD(m, n) = SAD(m, n) + |cur(i, j) - ref(i + m, j + n)|$ 
9:       end for
10:    end for
11:    if  $SAD(m, n) < SAD_{min}$  then
12:       $SAD_{min} = SAD(m, n)$ ;
13:       $MV = (m, n)$ ;
14:    end if
15:  end for
16: end for

```

For example, the computational complexity required to perform ME in real time for a video sequence in Common Intermediate Format (CIF) (352×288 @ 30 fps) and for a search range of size $[-16, 15]$ is 9.3 Giga Operations per Second (GOPS). If the frame size becomes DVD format (720×480 @ 30 fps) and the searching range is increased to $[-32, 31]$, the required computational complexity also increases to 127 GOPS. This extremely large computational complexity for full search based ME has motivated the development of many fast search algorithms. In the next subsections, several fast search algorithms will be discussed.

2.1.2 Fast Search Algorithms for Block Matching Algorithm

In order to reduce the huge computational requirement for motion estimation based on full search algorithm, a large number of fast but sub-optimal BMAs can be found in the literature [6–11]. These algorithms reduce the computational time as well as the hardware overhead to a considerable extent. However, the major drawback associated with these fast search algorithms is that very often they may be trapped in some local minima and thereby produce suboptimal results. Fast search algorithms can be broadly classified into three categories, namely (i) reduction in the number of search candidates [11–15] (ii) exploiting different matching criteria instead of the classical SAD [16–18], and (iii) predictive search [19–22] based on their characteristics.

In the following subsection, a brief introduction to these categories and some typical examples are presented.

2.1.2.1 Reduction in the Number of Search Candidates

These algorithms are based on the assumption that the distortion monotonically decreases as the search candidate approaches the optimal one. That is, even if all the search candidates are not matched, the optimal search candidate can be obtained by following the search candidate with the smallest distortion. This category accounts for the majority of fast search algorithms, and there are many algorithms available in literature, such as three step search [12], Successive Elimination, cross search [13], new-three-step search [6], four step search [12], unrestricted center-biased diamond search [23], diamond search [9], and so on.

Figure 2.2 depicts the search process for the Three Step Search (TSS) [12]. This algorithm was introduced by Koga et al. [12]. It became very popular because of its simplicity. It searches for the best motion vectors in a coarse-to-fine search pattern. In the first step, an initial step size is fixed. Eight blocks at a distance of the step size from the center (around the center block) are picked for comparison. In the next step, the center is moved to the point giving the minimum distortion with the step size halved. This is repeated till the step size becomes smaller than 1. One problem that occurs with TSS is that, it uses a uniformly allocated checking point pattern in the first step, which becomes inefficient for small motion estimation.

In Successive Elimination algorithm, motion vector for each reference block in the current frame can be find with much less computational load than exhaustive search algorithm by using some mathematical properties, which is discussed in Chap. 3.

Fig. 2.2 The search process for three step search

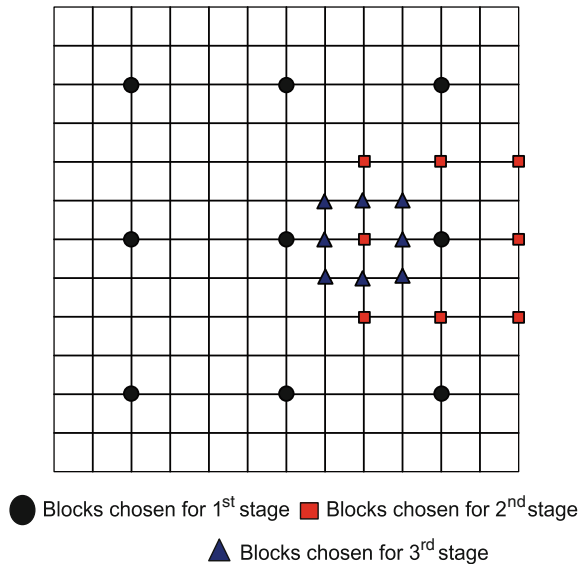
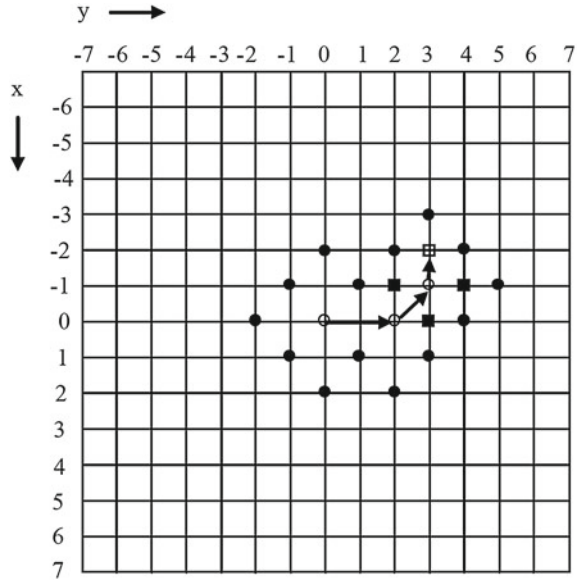


Fig. 2.3 The search procedure for diamond search algorithm



Diamond Search (DS) [9] is another typical fast search algorithm and is shown in Fig. 2.3 DS has two search steps namely, large diamond and small diamond. In the searching procedure, the large diamond step is applied first. DS continues in the large diamond step until the search candidate at the center has the smallest distortion among the nine candidates of the large diamond. Next, the small diamond is used to refine the searching result of the large diamond. Figure 2.3 portrays an example of DS algorithm. The arrow is the direction toward which the large diamond moves, and after the searching result of the large diamond converges, the small diamond is adopted to refine the searching result in the last step.

2.1.2.2 Simplification of Matching Criteria

The matching criterion of the block matching ME method has a direct impact on the coding efficiency and the computational complexity. Many matching criteria have been proposed in literature e.g., mean square error, SAD, pel difference classification etc. [5]. Whatever may be the matching criterion, evaluation of the matching criterion on pixels with 8 bits/pixel representation requires a huge amount of computation. The computational load can be reduced to a great extent by representing the pixels with a reduced number of bits. This method is known as pixel truncation. As proposed in [24], the number of bits in each pixel is truncated to achieve the reduction in computation. For example, if the number of bits in each pixel is truncated from eight bits to five bits, then the required computational load is only 5/8 of the original. Moreover, not only does pixel truncation serve to reduce the computational complexity, it also saves the hardware cost and the power consumed by ME hardware.

This is because a subtractor with less bit width can be used instead of that with eight bits. In the majority of video sequences, any pixel can be truncated to only six or five bits without much degradation in the quality.

It has been shown in [25] that for motion estimation based on pixel truncation, optimum results may be obtained by using Difference Pixel Count (DPC) as the matching criteria instead of the conventional SAD. For a block of size $N \times N$ and for a search range $[-p, p - 1]$, the DPC at any location (m, n) can be found as [25]:

$$DPC(m, n) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \bar{\delta}[\hat{C}(i, j), \hat{R}(i + m, j + n)] \quad (2.2)$$

Here, $\hat{C}(i, j)$ and $\hat{R}(i + m, j + n)$ represent bit truncated values for the pixels from the CB and the SW respectively. In Eq. 2.2, $\bar{\delta}(x, y)$ represents the standard delta function, for which $\bar{\delta}(x, y) = 0$ if $(x = y)$; else its value is 1.

In another method, as proposed in [16], an image frame with 8 bits/pixel representation is first converted into a binary frame with 1 bit/pixel representation. Motion estimation is then carried out on these binary image frames. Boolean exclusive OR (XOR) operation is used to find the Number of Non-Matching Points (NNMP), which is used as the matching criterion in place of the conventional SAD. The NNMP at any point (m, n) for a MB of size $N \times N$ is found as:

$$NNMP(m, n) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} B^t(i, j) \oplus B^{t-1}(i + m, j + n) \quad (2.3)$$

where, $-s \leq m, n \leq s$

Here, s is the maximum search range and \oplus denotes the XOR operation. Also, B^t and B^{t-1} represent the current and the reference 1BT frames respectively.

2.1.2.3 Predictive Search

The main problem associated with the fast search algorithms is that they are usually trapped into a local minimum. In order to avoid this condition, predictive search is developed and combined with other fast search algorithms. The concept of predictive search is based on the assumption that the Motion Vectors (MVs) of neighboring MBs are correlated and similar, so that they can be used to predict the MV of the current MB. Besides the spatial information, the temporal information also can be used in the process of prediction because of the motion continuity in the temporal direction. Therefore, the motion information of neighboring blocks in the spatial or temporal space is used to serve as the initial search candidate of fast search algorithms instead of the original point.

As proposed in [15], the initial search candidate can be the MVs of the blocks on the top, left, and top-right, their median, zero MV, the MV of the collocated block

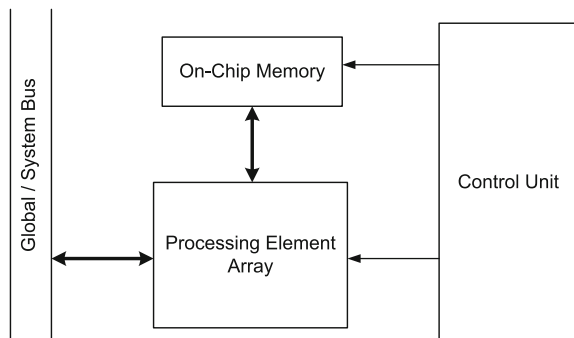
in the previous frame, and the accelerated motion vector of the collocated block in the previous two frames. By this way, the searching range can be reduced and constrained, so that not only the computational complexity but also the bit-rate of MV can be reduced. The Fast three step search algorithm (FTSS) [26] makes use of the directional information from adjacent previous motion vector and unimodal error surface assumption (UESA). It determines the direction of the current motion vector from the previous motion vector and reduces the computation for checking the candidate motion vector using the UESA. The UESA means that the error increases monotonically in getting away from the global minimum [27].

2.1.3 Motion Estimation Architectures

In order to achieve real time computation of Motion Estimation (ME), it is required that the ME hardware should be fast and at the same time should consume low power. Many ME architectures have been proposed in the last few years. In general, ME hardware can be broadly divided into two parts, the Processing Element (PE) array and the on-chip memory, as shown in Fig. 2.4.

PE array is the major operational core and responsible for the computation of SAD as given by Eq. 2.1. Although motion estimation involves simpler arithmetic computations, it involves a huge amount of memory access which involves considerable power consumption and also affects the overall speed of operation [24]. The required data are loaded through a global data bus. Moreover, in order to reduce the required memory bandwidth, some of the data are stored in the on-chip memory for data re-use. For each Macroblock (MB), the PE array gets the required data from both the global/system bus and the on-chip memory, and computes the corresponding SADs. At the same time, the data in the on-chip memory are updated for data re-use of the next MB or the search candidate. Depending on different ME algorithms and characteristics of PE array, the ME architectures can be broadly classified into three types: inter-level, intra-level, and tree-based architectures. While all inter-level and intra-level architectures have been used mostly for implementing full search algorithm,

Fig. 2.4 The block diagram of a typical ME architecture



tree-based architectures have been used for realizing fast search algorithms. In the following subsections, a brief survey of ME architectures are presented.

2.1.3.1 Motion Estimation Architectures for Full Search Algorithm

The full search algorithm can provide the best quality among various ME algorithms, but involves a huge amount of computation. Although the computational requirement for full search is large, it is preferred for VLSI implementation because of its simple operations and regular data flow compared to the fast search algorithms. Many different types of architectures have been proposed for the full search algorithm. The inter-level and intralevel architectures are the most commonly used. The first VLSI implementation of motion estimator was due to Yang et al. [28], who implemented the ME architecture for full search algorithm. This architecture was based on inter-level ME architecture. The PE in inter-level architectures is responsible for the computation of one search location. One PE computes the differences of all the pixels in the current block and accumulates the SAD pixel by pixel. The partial SAD is stored in each PE, until the SAD calculation of one search location is finished. A comparator is responsible for selecting the minimum SAD among all the generated SADs.

A detailed systolic mapping procedure to derive full search BMA architectures was proposed by Komarek and Pirsch [29]. This architecture was based on intra-level architecture. In an intra-level architecture, current pixels are stored in the corresponding PEs, and the reference pixels are propagated from one PE to another. The PE is responsible for calculating the distortion between one specific current pixel and the corresponding reference pixel for all the search candidates. In each PE, the distortion of a current pixel in current MB is computed and added to the partial SAD, which is propagated from the other PEs. After the initial cycles (depending upon the block size), the SADs are generated one by one and the comparator selects the minimum of them.

A large number of architectures have been proposed based on these two basic architectures. For example, the extension of architecture [28] has been proposed in [30], and besides one-dimensional inter-level architectures, two two-dimensional inter-level semisystolic architectures have been proposed in [31, 32]. The architecture [33] is an extension of [29]. The architectures described in [34] and [35] are two other intra-level architectures with large registers for fewer data inputs and memory bandwidth.

2.1.3.2 Motion Estimation Architectures for Fast Search Algorithms

Unlike the full search algorithm, the data flow for fast search algorithms is irregular and the processing order of the search candidates is not predefined. Rather, it depends on the last searching result. Thus, although the computational complexity of fast search algorithms is much smaller than that of the full search algorithm, the irregular data flow required for the fast search algorithms poses considerable challenge for

VLSI implementation. This makes the implementation of fast search algorithms much more difficult than that of the full search algorithm. Jong et al. [36] developed a fully pipelined parallel architecture for three step search BMA. Basically, 9 PEs compute the SAD of nine candidates in each step, and 256 cycles are required in each of the three steps. An intelligent data and memory arrangement are used to utilize the advantage of three step search procedure. Tree-based architectures developed by Jong et al. have the advantages of short latency, support for random access of the search candidates, and absence of pipelining bubbles cycles. A tree-based architecture that can support DS and fast full search algorithms has been proposed in [37]. As shown in Fig. 2.3, there are many duplicated search candidates between two successive steps in DS. After each search step for large diamond pattern, only five or three search candidates need to be calculated, and the others are calculated at the last large diamond pattern. In order to avoid the duplicated search candidates, a ROM-based solution, which uses a ROM to check if the search candidate is required to be computed or not, has been proposed in [37]. As stated in [37], the ROM-based solution can save 24.4 % search candidates in the DS algorithm, and the area overhead is also not significant. This architecture also supports fast full search algorithms.

Several architectures have been proposed for fast search algorithms besides tree-based architectures. For example, Dutta and Wolf [38] have modified the data flow of the 1-D linear array in [28] to support full search, TSS, and the conjugate direction search in the same architecture. A joint algorithm architecture design of a programmable motion estimator chip has been proposed by Lin et al. [39]. Cheng and Hang have proposed architecture based on universal systolic arrays to realize many BMAs [40].

2.2 Scalable Video Coding

Scalable Video Coding (SVC) must support more than one spatial, temporal and quality layer; hence the Codec structure of SVC differs from the conventional hybrid video coding structure of the H.264 video standard. There have been many contributors to the codec structure for an SVC. The first such contribution to the SVC codec structures was by Ohm [41]. Some of his video codec designs were modifications of the conventional hybrid coding [42–46]. Hybrid coding has been prevalent in all the video coding standards since the introduction of motion compensation for video coding. In case of encoding of the hybrid video structure, one frame predicts another, the predicted frame predicts another and this goes on for a GOP. It is quite evident and stated in [47] that prediction error tends to accumulate and the quality of the frames worsens as we move towards the last frame of the GOP. To avoid such a problem Motion Compensated Temporal Filtering has been introduced [41, 48–50].

Application of Motion Compensation (MC) is a key for high compression performance in video coding, but still is often understood to be implicitly coupled with frame prediction schemes. There is indeed no justification for this restriction, as MC can rather be interpreted as a method to align a filtering operation along the

temporal axis with a motion trajectory. In the case of MC prediction, the filters are in principle linear predictive coding (LPC) analysis and synthesis filters, while in cases of transform or wavelet coding, transform basis functions extended over the temporal axis are subject to MC alignment. This is known as motion-compensated temporal filtering. If MCTF is used in combination with a 2-D spatial wavelet transform, this shall be denoted as a 3-D or (depending on the sequence of the spatial and temporal processing) either as a 2-D+t or t+2-D wavelet transform. In case of MCTF as shown in Fig. 2.5, the error frame is used to update the reference frame; hence the error remains within the candidate and reference frames and is not accumulated or propagated to the successive frames.

Andreopoulos et al. [48, 49], introduced Wavelet-Based Scalable Video Coding. Instead of the conventional DCT, DWT was introduced as a suitable Image and Video Transform. Discrete Wavelet Transform is a multi-resolution transform. By using this property, DWT provides for an improved representation of the digital video data in a hierarchical manner. The latter being a useful property which can be extensively used in case of SVC. First codec structure which introduces the property of MCTF was SD-MCTF [48]. SD-MCTF perform the motion compensation in temporal axis on the pixel domain. The residual frames were then spatially decomposed using a suitable Discrete Wavelet Filter. For better coding efficiency [41], IB-MCTF [49] was introduced. Because the motion compensation was performed in the wavelet domain, the problem of shift-variance will be seen. In order to solve this problem, Over-complete DWT [48, 49, 51, 52] has to be performed. The algorithm and complexity models of IB-MCTF were shown in [49, 51]. Conventionally wavelet decomposition was performed by the convolution method. The problem with convolution based DWT was higher memory requirement and greater computational time. To avoid this problem a mathematical approach called Lifting scheme was introduced. Factorization of Discrete Wavelet Filters was done by Daubechies and Sweldens [53] and the results have been used in the implementation of our architecture. Details of over complete DWT (ODWT), lifting based ODWT, SD-MCTF, and IB-MCTF are given in the Chap. 7.

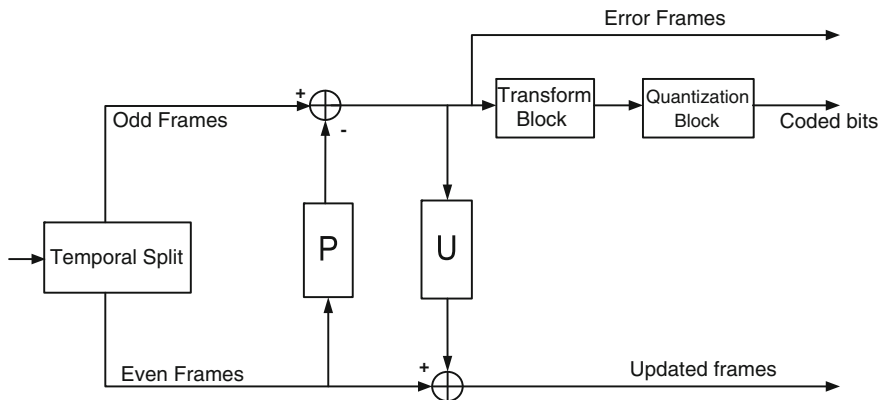


Fig. 2.5 Compensated temporal filtering block with predict and update stage

2.3 Conclusions

This chapter has presented the basic elements of ME algorithms and architectures which will be used in the following chapters. The concepts of motion estimation, block matching algorithm, different ME algorithms, ME architectures, challenges in hardware implementation and advantages of fast search algorithms have been discussed in this chapter. The ME algorithms have been classified into two categories, namely full search and fast search algorithms. Fast search algorithms are further classified based on simplification of matching criteria, reduction of search candidates, and predictive search. The ME architectures on the other hand, are separated into two parts, namely the processing element array and the on-chip memory. In the processing element array, inter-level, intra-level, and tree-based architectures are three basic types of ME architectures and are adopted in many ME architectures and video coding systems.

Scalable video coding based on IB-MCTF has a better PSNR performance than SD-MCTF because there is more degree of freedom of choosing in the ME schemes for the different sub-bands. But the memory requirement and the computational complexity increase as we go down higher levels of spatial and temporal scalability. IB-MCTF is preferred in research oriented fields where the quality of the received video data is more significant. In domains of medical imaging and distant medical applications where the quality of the video is significantly more important than the end-to-end delay, IB-MCTF is better than SD-MCTF.

In the next chapter, an implementation of one of the most popular motion estimation algorithm, namely the Fast Three step search algorithms will be discussed.

References

1. Biemonda, J., Looijengaa, L., Boekeea, D.E., Plompenb, R.H.J.M.: A pel-recursive Wiener-based displacement estimation algorithm. *J. Signal Process.* **13**(4), 399–412 (1987)
2. Wiegand, T., Sullivan, G.J., Bjontegaard, G., Luthra, A.: Overview of the H.264/AVC video coding standard. *IEEE Trans. Circuits Syst.* **7**(7), 560–576 (2003)
3. Lu, J., Liou, M.L.: A simple and efficient search algorithm for block-matching motion estimation. *IEEE Trans. Circuits Syst. Video Technol.* **7**(2), 429–433 (1997)
4. Tekalp, A.M.: *Digital Video Processing*. Prentice Hall Ltd., New York (1995)
5. Ghanbari, M.: *Video Coding, An Introduction to Standard Codecs*. The Institute of Electrical Engineers, London (1999). Chaps. 2, 5, 6, 7 and 8.
6. Li, R., Zeng, B., Liou, M.L.: A new three-step search algorithm for block motion estimation. *IEEE Trans. Circuits Syst. Video Technol.* **4**(4), 438–442 (1994)
7. Po, L.-M., Ma, W.-C.: A novel four step search algorithm for fast block motion estimation. *IEEE Trans. Circuits Syst. Video Technol.* **6**(3), 313–317 (1996)
8. Li, W., Salari, E.: Successive elimination algorithm for motion estimation. *IEEE Trans. Image Process.* **4**(1), 105–107 (1995)
9. Zhu, S., Ma, K.K.: A new diamond search algorithm for fast block-matching motion estimation. *IEEE Trans. Image Process.* **9**(2), 287–290 (2000)
10. Nie, Y., Ma, K.K.: Adaptive rood pattern search for fast block-matching motion estimation. *IEEE Trans. Image Process.* **11**(12), 1442–1448 (2002)

11. Cheung, C.H., Po, L.M.: A novel cross-diamond search algorithm for fast block motion estimation. *IEEE Trans. Circuits Syst. Video Technol.* **12**(12), 1168–1177 (2002)
12. Koga, T., Iinuma, K., Hirano, A., Iijima, Y., Ishiguro, T.: Motion compensated interframe coding for video conferencing, *Proceedings of Natural Telecommunication Conference*, pp. C9.6.1-C9.6.5, (1981)
13. Ghanbari, M.: The cross search algorithm for motion estimation. *IEEE Trans. Commun.* **38**(7), 950–953 (1990)
14. Chen, L.G., Chen, W.T., Jehng, Y.S., Chiueh, T.D.: An efficient parallel motion estimation algorithm for digital image processing. *IEEE Trans. Circuits Syst. Video Technol.* **1**(4), 378–385 (1991)
15. Tourapis, A.M., Au, O.C., Liu, M.L.: Highly efficient predictive zonal algorithms for fast block-matching motion estimation. *IEEE Trans. Circuits Syst. Video Technol.* **12**(10), 934–947 (2002)
16. Natarajan, B., Bhaskaran, V., Konstantinides, K.: Low-complexity block-based motion estimation via one-bit transforms. *IEEE Trans. Circuits Syst. Video Technol.* **7**(3), 702–706 (1997)
17. Liu, B., Zaccarin, A.: New fast algorithms for the estimation of block motion vectors. *IEEE Trans. Circuits Syst. Video Technol.* **3**(2), 148–157 (1993)
18. Wang, Y., Wang, Y., Kuroda, H.: A globally adaptive pixel decimation algorithm for block-motion Estimation. *IEEE Trans. Circuits Syst. Video Technol.* **10**(6), 1006–1011 (2000)
19. Hsieh, C.H., Lu, P.C., Shyn, J.S., Lu, E.H.: Motion estimation algorithm using interblock correlation. *IEEE Electron. Lett.* **26**(5), 276–277 (1990)
20. Zafar, S., Zhang, Y.Q., Baras, J.S.: Predictive block matching motion estimation for TV coding-part I: inter-block prediction. *IEEE Trans. Broadcast.* **37**(3), 97–101 (1991)
21. Zhang, Y.Q., Zafar, S.: Predictive block-matching motion estimation for TV coding-part II: inter-frame prediction. *IEEE Trans. Broadcast.* **37**(3), 102–105 (1991)
22. Chalidabhongse, J., Kuo, C.C.J.: Fast motion vector estimation using multiresolution-spatio-temporal correlations. *IEEE Trans. Circuits Syst. Video Technol.* **7**(3), 477–488 (1997)
23. Tham, J.Y., Ranganath, S., Ranganath, M., Kassim, A.A.: A novel unrestricted center biased diamond search algorithm for block motion estimation. *IEEE Trans. Circuits Syst. Video Technol.* **8**(4), 369–377 (1998)
24. He, Z.L., Tsui, C.Y., Chan, K.K., Liou, M.L.: Low-power VLSI design for motion estimation using adaptive pixel truncation. *IEEE Trans. Circuits Syst. Video Technol.* **10**(5), 669–678 (2000)
25. Lee, S., Kim, J.M., Chae, S.I.: New motion estimation algorithm using adaptively quantized low bit-resolution image and its VLSI architecture for MPEG2 video encoding. *IEEE Trans. Circuits Syst. Video Technol.* **8**(6), 734–744 (1998)
26. Kim, J.-N., Choi, T.-S.: A fast three step search algorithm with minimum checking points. In: *Proceedings of IEEE Conference Consumer Electronics*, pp. 132–133. 2–4 June 1998
27. Jianjua, L., Liou, M.L.: A simple and efficient search algorithm for block-matching motion estimation. *IEEE Trans. Circuits Syst. Video Technol.* **7**(2), 429–433 (1997)
28. Yang, K.M., Sun, M.T., Wu, L.: A family of VLSI designs for the motion compensation block-matching algorithm. *IEEE Trans. Circuits Syst.* **36**(2), 1317–1325 (1989)
29. Komarek, T., Pirsch, P.: Array architectures for block matching algorithms. *IEEE Trans. Circuits Syst.* **36**(2), 1301–1308 (1989)
30. Shen, J.F., Wang, T.C., Chen, L.G.: A novel low-power full search blockmatching motion estimation design for H.263+. *IEEE Trans. Circuits Syst. Video Technol.* **11**(7), 890–897 (2001)
31. Yeo, H., Hu, Y.H.: A novel modular systolic array architecture for full-search block matching motion estimation. *IEEE Trans. Circuits Syst. Video Technol.* **5**(5), 407–416 (1995)
32. Lai, Y.K., Chen, L.G.: A data-interlacing architecture with two dimensional data reuse for full-search block-matching algorithm. *IEEE Trans. Circuits Syst. Video Technol.* **8**(2), 124–127 (1998)
33. Chang, S.F., Hwang, J.H., Jen, C.W.: Scalable array architecture design for full search block matching. *IEEE Trans. Circuits Syst. Video Technol.* **5**(4), 332–343 (1995)

34. Vos, L.D., Stegherr, M.: Parameterizable VLSI architectures for the full-search block-matching algorithm. *IEEE Trans. Circuits Syst.* **36**(2), 1309–1316 (1989)
35. Roma, N., Sousa, L.: Efficient and configurable full-search blockmatching processors. *IEEE Trans. Circuits Syst. Video Technol.* **12**(12), 1160–1167 (2002)
36. Jong, H.-M., Chen, L.-G., Chiueh, T.-D.: Parallel architectures for 3-step hierarchical search block-matching algorithm. *IEEE Trans. Circuit Syst. Video Technol.* **4**(4), 407–416 (1994)
37. Chao, W.M., Hsu, C.W., Chang, Y.C., Chen, L.G.: A novel motion estimator supporting diamond search and fast full search. In: *Proceedings of IEEE International Symposium Circuits Systems (ISCAS02)*, pp. 492–495 (2002)
38. Dutta, S., Wolf, W.: A flexible parallel architecture adopted to block matching motion estimation algorithms. *IEEE Trans. Circuits Syst. Video Technol.* **6**(1), 74–86 (1996)
39. Lin, H.D., Anesko, A., Petryna, B.: A 14-GOPS programmable motion estimator for H.26X video coding. *IEEE J. Solid-State Circuits* **31**(11), 1742–1750 (1996)
40. Cheng, S.C., Hang, H.M.: A comparison of block-matching algorithms mapped to systolic-array implementation. *IEEE Trans. Circuits Syst. Video Technol.* **7**(5), 741–757 (1997)
41. Ohm, J.-R.: Advances in scalable video coding. *Proc. IEEE, Invit. Pap.* **93**, 42–56 (2005)
42. Advanced Video Coding for Generic Audiovisual Services, ITU-T Rec. H.264 and ISO/IEC 14496-10 (MPEG-4 AVC), ITU-T and ISO/IEC JTC 1, Version 1: May 2003, Version 2: May 2004, Version 3: March 2005, Version 4: September 2005, Version 5 and Version 6: June 2006, Version 7: April 2007, Version 8 (including SVC extension): Consented in July 2007
43. ITU-T Rec. & ISO/IEC 14496-10 AVC.: Advanced video coding for generic audiovisual services, version 3 (2005)
44. Schwarz, et al. H.: Technical description of the HHI proposal for SVC CE1. ISO/IEC JTC1/WG11, Doc. m11244, Palma de Mallorca, Spain, October 2004
45. Reichel, J., Schwarz, H., Wien, M.: Scalable video coding joint draft 6 Joint video team, Doc. JVT-S201, Geneva, Switzerland, April 2006
46. Coding of audiovisual objectsPart 10: Advanced video coding, International Organization for Standardization/International Electrotechnical Commission (ISO/IEC), ISO/IEC14 496–10 (identical to ITU-T Recommendation H.264)
47. Richardson, I.E.G.: *Video Codec Design*, John Wiley & Sons. Ltd. (2002). doi:[10.1002/0470847832](https://doi.org/10.1002/0470847832)
48. Andreopoulos, Y., van der Schaar, M., Munteanu, A., Barbarien, J., Schelkens, P., Cornelis, J.: Complete-to-overcomplete discrete wavelet transforms for scalable video coding with MCTF. In: *Proceedings SPIE/IEEE Visual Communication Image Process.*, pp. 719–731 (2003)
49. Andreopoulos, Y., Munteanu, A., Barbarien, J., van der Schaar, M., Cornelis, J., Schelkens, P.: In-band motion compensated temporal filtering. *Signal Process. Image Commun.* **19**, 653–673 (2004)
50. Wang, B., Loo, K.K., Yip, P.Y., Siyau, M.F.: A simplified scalable wavelet video codec with MCTF structure. In: *International Conference on Digital Telecommunications, (ICDT'06)*, 29–31 August 2006. doi:[10.1109/ICDT.2006.11](https://doi.org/10.1109/ICDT.2006.11)
51. Andreopoulos, Y., van der Schaar, M., Munteanu, A., Barbarien, J., Schelkens, P., Cornelis, J.: Fully-scalable wavelet video coding using in-band motion-compensated temporal filtering. In: *Proceedings IEEE International Conference Acoustical Speech and Signal Process.*, pp. III-417–III-420 (2003)
52. Park, H.-W., Kim, H.-S.: Motion estimation using low-band-shift method for wavelet-based moving-picture coding. *IEEE Trans. On Image Process.* **9**(4), 577–587 (2000)
53. Daubechies, I., Sweldens, W.: Factoring wavelet transforms into lifting steps. *J. Fourier Anal. Appl.* **4**(3), 247–269 (1998)

Motion Estimation for Video Coding

Efficient Algorithms and Architectures

Chakrabarti, I.; Batta, K.N.S.; Chatterjee, S.K.

2015, XVIII, 157 p. 67 illus., Hardcover

ISBN: 978-3-319-14375-0