Proceedings of the 3rd International Conference on Signal Processing and Machine Learning
DOI: 10.54254/2755-2721/6/20230847

Motion estimation algorithm and architecture survey

Shengyang Chen

Shenzhen University, 3688 Nanhai Avenue, Nanshan District, Shenzhen, Guangdong Province, 518060, China

shengyang.chen@foxmail.com

Abstract. Motion estimation is a key part of video temporal characteristics analysis including video filter and compression. Increasing requirements for high-quality video image processing have confirmed the necessity of researching motion estimation. Current motion estimation methods, including high-efficient fast motion estimation algorithms and hardware architecture designs, achieved different performances from different perspectives. In this paper, we focus on analyzing typical fast algorithms and hardware architecture designs for motion estimation, comparing the used methods, and pointing out the basic ideas of motion estimation research schemes. In terms of theoretical implications, the research contributed to a more comprehensive reference for fast motion estimation algorithms and their hardware architecture designs.

Keywords: Motion estimation, Fast block-matching algorithm, H.264/AVC, HEVC, Hardware architecture.

1. Introduction

Motion estimation (ME) is a motion information extraction process, it plays a key role in many fields such as video coding/compression, and temporal filtering. To attain high coding efficiency, ME is essential in video coding algorithms that can successfully reduce temporal redundancies between neighboring video frames. The Block Matching based Algorithm (BMA) algorithm is one of the most common motion ME methods including the Pixel Matching algorithm, Region matching algorithm, and so on. BMA is the most widely used due to its high matching accuracy and efficiency, which will help to achieve high video coding efficiency. In this paper, we mainly focus on BMA ME applied in video coding.

Typical BMA consists of a Full Search(FS)[1], Three-Step Search(TSS)[2], New Three-Step Search (NTSS)[3], Four-Step Search(FSS)[4], Diamond Search(DS)[5], and TZsearch(TS)[6]. FS is the most accurate algorithm but it also costs a large amount of search time. TSS uses one rectangle search pattern that only needs to search 25 points, it reduces the computation complexity compared to FS, but the search strategy makes it possible to fall into a local optimum. NTSS improves TSS results by providing a center-based searching scheme and involves provisions for a halfway stop to reduce computational costs. NTSS also reduces the probability of falling into a local minimum. But it has poor performance when processing intense motion images. FSS is similar to TSS, while a more flexible search pattern is proposed. This algorithm has more computation complexity but it avoids making mistakes in search directions. Two search patterns including the large diamond search pattern to avoid falling into a local optimum and the small diamond search pattern to avoid searching path errors and achieves better

DOI: 10.54254/2755-2721/6/20230847

performance are employed in DS. To further reduce complexity and improve accuracy, TS employs variable step search patterns and initial point prediction methods, which have significantly improved search accuracy and performance as an application tool in modern video coding software including HM16.0 for H.265/HEVC [7]. However, TS, as a fast adaptive algorithm, has a close dependency between adjacent steps that are not conducive to hardware implementation, and irregular search points bring inconvenience to pixel data acquisition.

Although these typical algorithms perform well, they have certain limitations in search speed, search performance, and hardware implementation that need to be improved and optimized. Typical ME algorithm optimization methods can be divided into two categories, to be more accurate or faster as high-performance search algorithms or fast search algorithms, respectively. However, they are usually optimized in different directions, which will limit the search accuracy or speed. Therefore, to optimize the ME performance or speed, researchers present various effective ME algorithm optimization methods instead of the traditional methods above.

In Djoudi Kerfa proposed a Star Diamond Search with Adaptive Threshold(SD_{th}) algorithm [8]. The initial search center prediction and early termination scheme of SD_{th} considerably reduce the computation complexity. Compared with DS, the SD_{th}employed a more efficient search pattern, which achieved higher accuracy with comparable speed. Quanyang Liu et al. proposed a novel fast motion estimation method[9], which presented a one-dimensional BMA ME algorithm based on the traditional ME algorithm. Different from other methods, it focused on dimensionality reduction to conduct the block-matching process. Experiments show that this algorithm has stable performance while processing high-quality videos. In Pavel Arnaudov et al. proposed an adaptive fast ME algorithm[10]. By dynamically changing the search pattern according to different situations, the computational complexity is considerably reduced.

In addition to optimizing ME from the perspective of the algorithm, there are many demands for hardware to realize motion estimation in practical applications. However, many optimizing algorithms are not suitable for hardware design such as [6], or with lower search performance as [2]. Hence researchers have presented some more efficient methods combining hardware design and algorithms.

Sushanta Gogoi et al. presented a hybrid algorithm with a TSMC 90 nm hardware architecture [11]. Its searching operations would begin from the large blocks to the small blocks in a hierarchical order to make better use of the data bandwidth. The result shows that it achieved better performance in resource consumption and coding time. In J. Zheng et al. presented a hardware-efficient motion estimation algorithm [12]. By applying a three-level memory organization, multiple search strategies, and early termination, this algorithm reduces the computation complexity while maintaining a stable coding performance.

An H.264 coding-based ME architecture was proposed in [13]. By logically applying parallel processing in the preprocessing and key point finding processes, this architecture implemented the hardware of ME. By filtering the informative data in the low-frequency range, the time-domain-based algorithm was changed into a frequency domain to decide the search range of ME. Experiments show that this algorithm and architecture significantly improved hardware implementation performance while maintaining signal quality.

In this paper, we review the typical ME optimization methods that have been proposed recently, and deeply analyze the characteristics and development trends of each method. In addition, we point out the possible ways to further improve search speed, accuracy, and hardware implementation of the ME algorithm. The remainder of this paper is organized as follows. Section II presents the recent research on ME algorithm optimization. Section III contains a detailed discussion regarding the hardware designs of ME of different algorithms. Finally, Section IV concludes this paper.

DOI: 10.54254/2755-2721/6/20230847

2. Typical Fast Motion Estimation Algorithm

2.1. The brife flow of motion estimation

In the process of BMA, each frame is divided into blocks according to certain rules to calculate the motion vector (MV). The motion estimation of the current block (CB) of the current frame is to find which candidate block is most similar to it in the previous frame. The basic procedure of BMA is: (1) Divide each frame into blocks that consist of $M * N$ pixels, where M and N denotes the block's width and height, respectively. (2) Determine the search range/window (3) Determine the best matching criteria (4) For each block in the current frame, the best matching block centered on its corresponding position $(x + u, y + v)$ in the reference frame should be searched (5) Get the horizontal and vertical displacement as the motion vector. Fig.1 shows the block-matching method based on motion estimation.

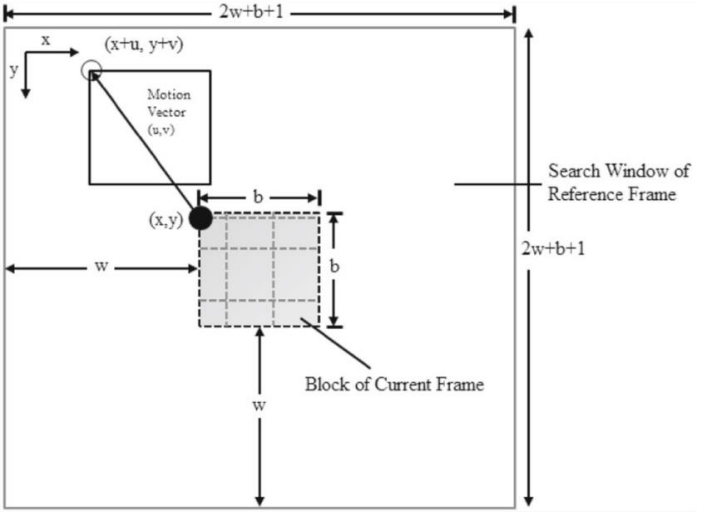

Figure 1. Block matching method [14].

Therefore, to find the best MV, we should focus on the search range, start searching point, search strategy, and matching criterion. For example, usually the FS starts searching from the center of the search window (SW) with a relative search range like [-256, 256]. The search strategy of FS is to search outward from the starting point in a clockwise direction and calculate the sum of the absolute difference (SAD) as the matching criterion of each search point as,

$$SAD(x,y) = \sum_{x=1}^{M} \sum_{y=1}^{N} |CF(\mathrm{x,y}) - RF(x+u, y+v)| \quad (1)$$

where CF is the pixel grey-value of (x,y) in the current frame, and RF represents the pixel grey-value of $(x + u , y + v)$ in the reference frame. The point with the minimum SAD is regarded as the best matching point. The vector difference between this point and the start point is taken as the motion vector. It can be seen that the FS can obtain the best search performance due to all candidate points in the SW are searched. However, FS involves large computation complexity, which results in great challenge for real time processing, especially for the hardware design. Hence efficient fast motion estimation algorithms and hardware architecture research are required.

DOI: 10.54254/2755-2721/6/20230847

2.2. Fast motion estimation algorithm by searching dimensionality reduction

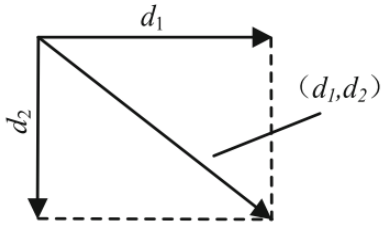

Figure 2. Motion vector decomposition [9].

Traditional two-dimensional BMA has a shortcoming in search speed. Researchers present a One-dimensional BMA (OBMA) to optimize the search efficiency [9]. The basic idea of OBMA is to calculate the MV step by step using horizontal and vertical motion vectors as shown in Fig.2. The specific calculation process is shown in Eq. (2),

$$\begin{cases} \hat{d}_1 = arg\min\limits_{d_1} \sum\limits_{n_1=0}^{N_1} \left| \sum\limits_{n_2=0}^{N_2} s(n_1, n_2, z) - \sum\limits_{n_2=0}^{N_2} s(n_1 + d_1, n_2, z+1) \right| \\ \hat{d}_2 = arg\min\limits_{d_2} \sum\limits_{n_2=0}^{N_2} \left| \sum\limits_{n_1=0}^{N_1} s(n_1, n_2, z) - \sum\limits_{n_1=0}^{N_1} s(n_1, n_2 + d_2, z+1) \right| \end{cases} \quad (2)$$

where $s(n_1, n_2, z)$ and $s(n_1 + d_1, n_2 + d_2, z + 1)$ represent the current and reference frame, respectively. n_1, n_2 , and z represents the coordinate value. N_1 and N_2 represents the maximum value of n_1 and n_2, respectively. Horizontal and vertical offset is respectively represented by d_1 and d_2. $\hat{d}_1$ and $\hat{d}_2$ represent the horizontal and vertical MV estimation values, respectively.

The search strategy of OBMA is TSS in one dimension. The search radius should be determined first to determine the search steps. There is a relationship between search steps (SN) and search radius (r): $r = 2^{SN} - 1$. Flexible setting different r according to the actual use of OBMA can reduce unnecessary SN, and improve search performance. The search steps of this strategy are as follows. (1) Determine the search steps, and set the middle point to zero value. (2) Search the center point and right and left points in radius steps. Calculate the minimum SAD of these points. Take the extreme point as the next search center. (3) Back to step2 until the search is finished. The last search center is the MV estimation value.

To estimate the coding efficiency, researchers use PNSR and coding time as the evaluation indicators. Under different video sequences and matching block sizes, the proposed method can achieve much less coding time than BMA while maintaining a steady search quality as BMA. However, when processing some other sequences, OBMA cannot guarantee that all of the coding performance is better than BMA. By comprehensive analysis, the overall efficiency of OBMA is better than BMA.

2.3. Quadrant-Oriented Fast Motion Search Algorithm

In [15], researchers proposed a new quadrant-oriented search algorithm (QSP) with a zero-motion prejudgment (ZMP) method. ZMP is applied to distinguish whether a block is moving or still and also reduces the computation complexity. The first step is to compute the block level distortion between CB and co-located block in the reference frame. Then they compare the distortion with a forecast threshold

DOI: 10.54254/2755-2721/6/20230847

value. If the threshold value is larger, the CB will be considered as a static block. Fig.3 shows the main process of the QSP.

Figure 3. Basic flow of the search strategy [15].

The accurate procedure is as follows. First, the ZMP technique for detecting non-motion blocks was applied. Then they calculate the SAD value of the center point in the QSP and compare it with the threshold value. If the threshold value is larger, take the point as the center and use the small DS pattern (SDSP) to search for a new minimum SAD value as the final value. If the threshold is smaller and the center point is placed on any one of the quantitative points, create a new QSP centered on the quantitative point with the minimum SAD value to calculate the final value, and ignore the candidate search points beyond the SW. In the processing of the algorithm, the search position is fixed in the quadrant according to the estimated SAD value and the search space in the SW is divided into equal quadrants. The search procedure is carried out until a perfect match is discovered. The size of the chosen SW determines the number of search steps in the search process. For both low and high speed video frames, the algorithm works best.

To estimate the performance of the algorithm, researchers use PSNR and motion estimation time as evaluation criteria compared with other algorithms such as the hexagon algorithm, DS, and adaptive root pattern algorithm. Experiments have confirmed that QSP has a higher PSNR value and less working time than other algorithms.

From the above discussion, the conclusion can be reached that by reducing the search points and search patterns, the computation complexity of motion estimation can be reduced efficiently.

3. Hardware Motion Estimation Architecture Design

The architecture design of motion estimation mainly focuses on reducing data dependency, storing data, and designing efficient parallel, pipelining hardware architectures to achieve different resolutions of real-time processing.

The challenges of motion estimation hardware architecture design mainly including: (1) Low search concurrency in the search range of the original software motion estimation algorithm, which limits the operation speed. (2) Unreasonable hardware pipelining design architectures with low throughput for the search process, which results in larger hardware areas. (3) Hard to balance many goals among the

DOI: 10.54254/2755-2721/6/20230847

performance of the motion estimation algorithm, hardware area, and processing speed to achieve cost-effective motion estimation hardware architecture.

Thus novel hardware architecture designs were proposed for these optimal goals. Two typical methods are given to demonstrate the considerations above.

3.1. A hybrid motion estimation algorithmwith hardware architecture design

By employing the square motion estimation method and hexagonal motion estimation method, a hybrid search pattern is presented in [11]. The start searching point is on the center and then spread different search patterns due to different situations. To find the best MV more effectively, this algorithm only searches for prediction blocks of certain sizes. It was intended to search every grid in SW in sequential order and run concurrent operations on every grid block. The algorithm has three steps.

Step1. Calculate the Rate-Distortion (RD) cost at origin (0,0) and the predicted MV positions to determine the start searching point.

Step2. Perform coarse search to find the best-matched block and calculate the RD cost. Compare the smallest RD cost with the start point. The point with less RD cost will be taken as the final point.

Step3. Perform refinement search. Applying different search patterns according to the location of the final point.

As for the architecture design, it was shown in Fig.4.

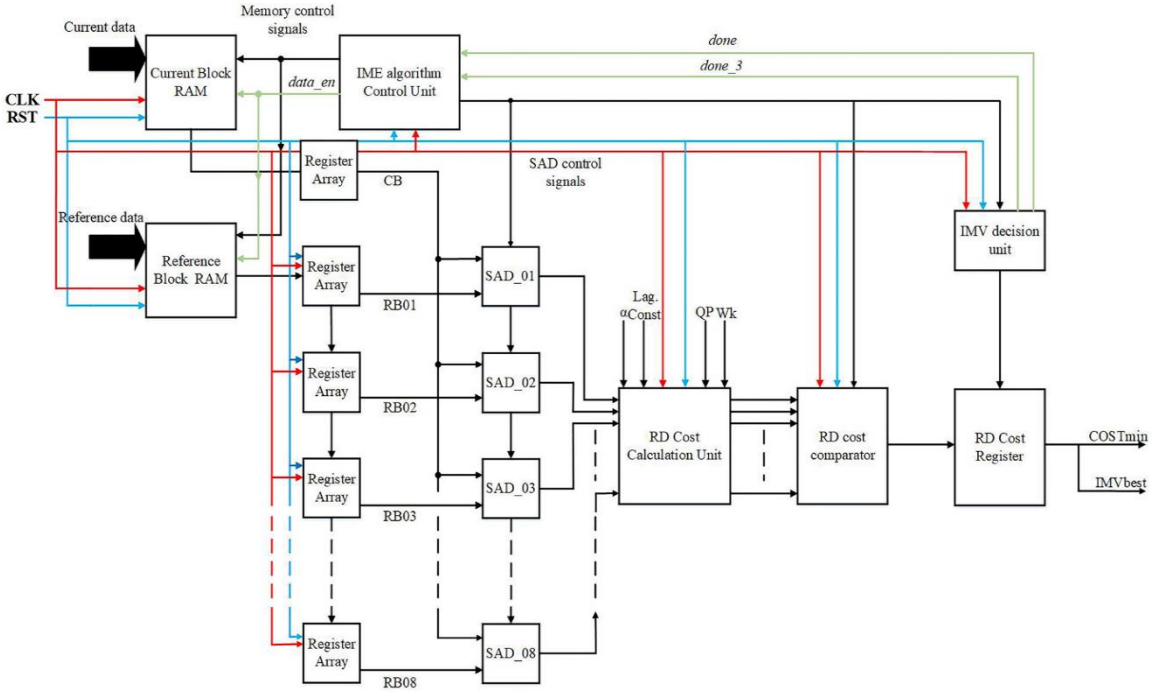

Figure 4. The overall hardware architecture for the algorithm [11].

The architecture is composed of five major units: (1) Register array unit, (2) IME algorithm control unit, (3) RD cost calculation unit, (4) IMV decision unit, and (5) SAD unit. For the Register array unit, its major function is to upload blocks of pixels from current and reference block memory. The IME control unit selects the memory blocks to be fed to the register array. RD cost unit is used for calculating the RD cost. By comparing each MV candidate's lowest RD cost, MV is determined by the IMV decision unit. SAD unit is applied to estimate the SAD value.

The architecture applied TSNC 90nm technology, its various data are as follows, the frequency is 162MHz, the memory is 18kB, the gate count is 2784.4K, the throughput is 2.78 Gpixels/s. the power consumption is 463.4 (334.5a) mW, and the resolution is 8192 × 4320 @78fps.

DOI: 10.54254/2755-2721/6/20230847

To verify the coding performance of the proposed algorithm, researchers tested different classes of video sequences compared to TS. The average coding time save of the proposed algorithm is 11.1952% and the average Y, U, V PSNR value is −0.1324,−0.1036, and −0.0805, respectively. The proposed algorithm has about the same average PSNR with TS in different resolutions, according to the results. What's more, the design achieved less coding time because of the new hardware's low search point count, high throughput, low processing clock cycle count, low memory usage, and ability to handle 8K HD films at a higher frame rate.

3.2. An efficient block level matched architecture hardware design and its supporting algorithm

J. Zheng et al. presented a hardware-efficient BMA and its hardware design [12]. The algorithm is based on three aspects including (1) predict the search center; (2) early termination, and (3) pixel down-sampling.

For the first aspect, the relative coordinate (centerx, centery) is defined by the following rules,

$$\begin{cases} center_x = median(CCSW_x, EX_x, UMVP_x, CCSW_x + EX_x) \\ center_y = median(CCSW_y, EX_y, UMVP_y, CCSW_y + EX_y) \end{cases} \tag{3}$$

where ($UMVP_x$, $UMVP_y$) is the uniform predicted MV and ($CCSW_x$, $CCSW_y$) is the center of current SW. For better motion estimation performance, the pixels in horizontal and vertical directions are extended by EX_y and EX_x, respectively. The accuracy of the search center is checked based on the SAD value estimated from current block and the block in the search center (SAD_{center})

During the search process, a dynamic threshold T_n is proposed to eliminate non-motion blocks. The threshold T_n is calculated by the following equation,

$$Tn = Min[Max(SADNon(avg)), 512), SAD_{center}] \tag{4}$$

where SADNon(avg) represents the average SAD value for all zero motion blocks that were coded before. If T_n is bigger than SADcenter, take the CB as a zero motion block and use the coordinate of the search center as the MV of CB. In the subsequent process, for hardware implementation, two parallel searches are performed. One is limited search in a limited search range to find major MVs, the other is hierarchy search which is applied to estimate the MVs of rapidly moving objects. The hierarchy search has two steps, namely the course search and refine search. In addition, for large-motion blocks, a wide search range [−128, +128) is covered by pixel down-sampling before ME, which can further reduce the computation complexity.

As to the hardware consideration to store reusable pixel data, the algorithm applied a three-level memory organization consisting of organized reg array, external storage, and on-chip static memory. It reduced the memory bandwidth requirement.

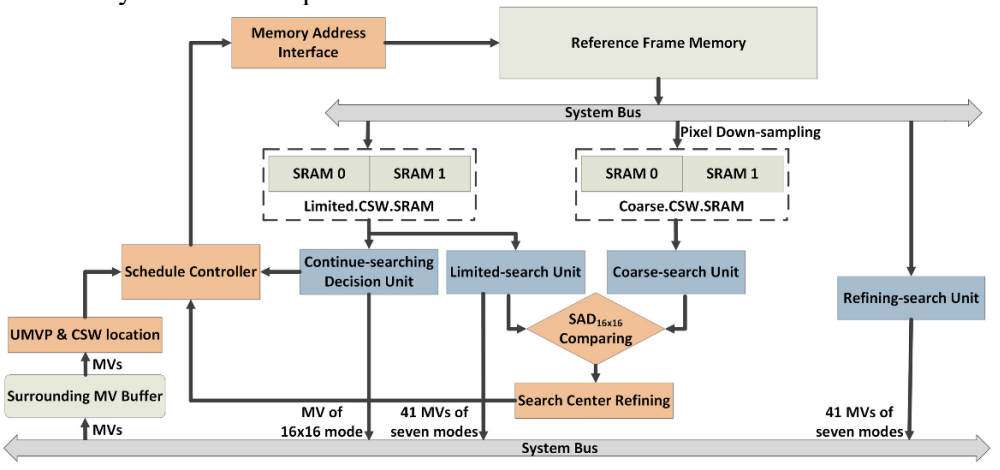

Figure 5. Flowchart of the efficient architecture design [12].

DOI: 10.54254/2755-2721/6/20230847

Fig. 5 is the flowchart of the hardware design. During the process, zero motion blocks will be eliminated by a Continue-searching Decision unit. Three search units work in parallel and play a major role. The Coarse-Search Unit uses pixel down-sampling to search blocks in a large motion while the Refining-Search Unit is responsible for refining its results. And the Limited-search Unit is used in the search center that was predicted before to search a small SR [−8, +8). The surrounding MV Buffer is used to estimate the UMVP value and CSW (Current SW) are stored in on-chip buffers by four different SRAMs as temporarily buffer reusable data in limited and coarse SRAMs which also improved the performance of parallel processing, respectively. What's more, the ping-pong rule (SRAM 0 and SRAM 1) is used for improving data loading efficiency.

By evaluated in the H.264 encoder JM18.0, the coding efficiency of the proposed algorithm was quantified. The proposed architecture used IBM45nm technology and its synthesized circuit is consist of 574K logic gates and on-chip static memory of nearly 20KB. Experiments show that the architecture achieved a Maximum throughput of 1162Mpixels/s at 500MHz for 2160p@30fps videos in a ± 128SR. the power consumption is 198mW while the quality loss is only -0.06dB, which is evaluated by BD-PSNR [16].

From the discussion, one may conclude that typical hardware design methods for motion estimation not only need to reduce the data dependency and processing complexity by improving motion estimation algorithms but also need to focus on designing efficient hardware storage structures and circuit microstructures at different levels.

4. Conclusion

This paper has presented a theoretical study of typical motion estimation algorithms and hardware architecture designs. The study analyzed the main process of some fast block-matching algorithms and the key microstructure design of the hardware structure in detail. Comparative analysis shows that fast motion estimation algorithms are usually optimized by reducing the search range, designing high-efficient search patterns, and searching for more accurate start-searching points. In the research of hardware design, the main optimization is to solve the complex data dependence and propose high-performance parallel and pipeline structures to achieve HD or UHD real-time video motion estimation performance. The current ME algorithm and its hardware design typically use empirical methods. Future research should consider the potential effects of machine learning methods more carefully, for example, combining deep learning to explore the relationship between complexity and accuracy of motion estimation.

References

[1] Morris, T., Britch, D.: Biorthogonal wavelets for intra-frame video coding. In: InternationalWorkshop on Image and Signal Processing and Analysis, pp. 209 – 214 (2000)

[2] Koga, T., Iinuma, K., Hirano, A., Iijima, Y., Ishiguro, T.: Motion compensated interframe coding for video conferencing. In: Proc. Nat. Telecommun. Conf. New Orleans, pp. 5.3.1 – 5.3.5 (1981)

[3] Brunig, M., Niehsen, W.: Fast full-search block matching. IEEE Trans. Circuits Syst. Video Technol. 11(2), 241 – 247 (2001)

[4] Po, L. M., & Ma, W. C. (1996). A novel four-step search algorithm for fast block motion estimation. IEEE transactions on circuits and systems for video technology, 6(3), 313-317.R. Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev., in press.

[5] Takamura, S., Yashima, Y.: H.264 based lossless video coding using adaptive transforms. In: Proceedings of IEEE ICASSP, pp. 301 – 304 (2005)

[6] R. Khemiri, N. Bahri, F. Belghith, F. E. Sayadi, M. Atri and N. Masmoudi, "Fast motion estimation for HEVC video coding," 2016 International Image Processing, Applications and Systems (IPAS), 2016, pp. 1-4, doi: 10.1109/IPAS.2016.7880120.

[7] Sullivan, G. J., Ohm, J. R., Han, W. J., & Wiegand, T. (2012). Overview of the high efficiency

DOI: 10.54254/2755-2721/6/20230847

video coding (HEVC) standard. IEEE Transactions on circuits and systems for video technology, 22(12), 1649-1668.

[8] Boonthep, N., Chamnongthai, K., & Phensadsaeng, P. (2020). H. 264 Video Coding-Based Motion Estimation Architecture for Video Broadcasting from a Studio. Wireless Personal Communications, 115(4), 2851-2874.

[9] Liu, Q., Liu, Y., Li, Q., Yan, F., Zhang, Q., Ma, Y., & Gao, W. (2022). One-dimensional block-matching motion estimation algorithm. Signal, Image and Video Processing, 1-9.

[10] Arnaudov, P., & Ogunfunmi, T. (2020). Dynamically adaptive fast motion estimation algorithm for HD video. Journal of Signal Processing Systems, 92(10), 1115-1131.

[11] Gogoi, S., & Peesapati, R. (2021). A hybrid hardware oriented motion estimation algorithm for HEVC/H. 265. Journal of Real-Time Image Processing, 18(3), 953-966.

[12] Zheng, J., Lu, C., Guo, J., Chen, D., & Guo, D. (2019). A hardware-efficient block matching algorithm and its hardware design for variable block size motion estimation in ultra-high-definition video encoding. ACM Transactions on Design Automation of Electronic Systems (TODAES), 24(2), 1-21.

[13] Boonthep, N., Chamnongthai, K., & Phensadsaeng, P. (2020). H. 264 Video Coding-Based Motion Estimation Architecture for Video Broadcasting from a Studio. Wireless Personal Communications, 115(4), 2851-2874.

[14] Manikandan, L. C., Nair, S. A. H., Sanal Kumar, K. P., & Selvakumar, R. K. (2019). A study and analysis on block matching algorithms for motion estimation in video coding. Cluster Computing, 22(5), 11773-11780.

[15] Shajin, F. H., Rajesh, P., & Raja, M. R. (2022). An efficient VLSI architecture for fast motion estimation exploiting zero motion prejudgment technique and a new quadrant-based search algorithm in HEVC. Circuits, Systems, and Signal Processing, 41(3), 1751-1774.

[16] G. Bjontegaard. 2001. Calculation of average PSNR differerces between RD-curves. In 13th VCEG-M33 Meeting. IUT-T,1 – 5.