International Scholarly Research Network
ISRN Signal Processing
Volume 2011, Article ID 386128, 10 pages
doi:10.5402/2011/386128

Research Article

A Fast Fractal Video Coding Algorithm Using Cross-Hexagon Search for Block Motion Estimation

Kamel Belloulata,[1] Shiping Zhu,[2] and Zaikuo Wang[2]

[1] *Department of Electronics, School of Engineering, University of Sidi Bel Abbès, Sidi Bel Abbès 22000, Algeria*
[2] *Department of Measurement Control & Information Technology, School of Instrumentation Science & Optoelectronics Engineering, Beihang University, Beijing 100191, China*

Correspondence should be addressed to Shiping Zhu, shiping.zhu@buaa.edu.cn

Received 8 March 2011; Accepted 28 April 2011

Academic Editor: A. Ito

We propose a novel fractal video coding method using fast block-matching motion estimation to overcome the drawback of the time-consuming character in the fractal coding. As fractal encoding essentially spends most time on the search for the best-matching block in a large domain pool, search patterns and the center-biased characteristics of motion vector distribution have large impact on both search speed and quality of block motion estimation. In this paper, firstly, we propose a new hexagon search algorithm (NHEXS), and, secondly, we ameliorate, by using this NHEXS, the traditional CPM/NCIM, which is based on Fisher's quadtree partition. This NHEXS uses two cross-shaped search patterns as the first two initial steps and large/small hexagon-shaped patterns as the subsequent steps for fast block motion estimation (BME). NHEXS employs halfway stop technique to achieve significant speedup on sequences with stationary and quasistationary blocks. To further reduce the computational complexity, NHEXS employs modified partial distortion criterion (MPDC). Experimental results indicate that the proposed algorithm spends less encoding time and achieves higher compression ratio and compression quality compared with the traditional CPM/NCIM method.

1. Introduction

Fractal image compression reduces the redundancy of images by using self-similarity properties and seems to be a favorable method for the image compression due to its advantages of high compression ratio, fast decompression, and resolution independence [1]. This is particularly suitable for the situation of one encoding and many decodings [2]. Fractal coding is thought publicly to be one of the three most developing outlook codec methods [3]. However, its major drawback is that fractal encoding is complex and time consuming to search for the best-matching block in a big pool of domain blocks, and this seriously embarrasses the fractal image coding method's application into the practice. Fisher classified the image blocks (range block and domain block) [1]. An image block was divided into four quadrants. The average and the variance were computed for each quadrant, so, for the four quadrants, 72 classes were constructed. The range block found matches only in the domain pool with the same class. This reduced the search space efficiently [3]. However, it needs a large amount of computations in the classifying process and some range blocks may not find the matching blocks in the same class in the domain pool [4]. As the developing of the fractal image compression, the fractal coding method has been applied in video sequence compression [2, 5], for instance, the famous hybrid circular prediction mapping (CPM) and noncontractive interframe mapping (NCIM) [6]. The CPM/NCIM combines the fractal coding algorithm with the well-known motion estimation and compensation (ME/MC) algorithm that exploits the high temporal correlations between adjacent frames. In CPM and NCIM, each range block is motion compensated by a domain block in the previous frame, which is of the same size as the range block even though the domain block is always larger than the range block in conventional fractal image codec. The main difference between CPM and NCIM

is that CPM should be contractive for the iterative decoding process to converge, while NCIM need not be contractive since the decoding depends on the already decoded frames and is noniterative [7].

Recently, Wang et al. [8] proposed a hybrid fractal video compression algorithm, which merges the advantages of a cube-based fractal compression method and a frame-based fractal compression method; in addition an adaptive partition instead of fixed-size partition is discussed. The adaptive partition [3] and the hybrid compression algorithm exhibit, relatively, high compression ratio for image [3] and the video conference sequences [8]. In conclusion, a fractal image codec performs better in terms of very fast decoding process as well as the promise of potentially good compression [4, 9–12]. But, at present, fractal codec is not standardized because of its huge calculation amount and slow coding speed. In order to alleviate the above difficulties, a novel fractal video coding algorithm using fast block-matching motion estimation technology [13, 14] is proposed in this paper to improve the encoding speed and the compression quality.

Block-matching motion estimation is a vital process for many motion-compensated and video coding standards. Motion estimation could be very computational intensive and can consume up to 60–80% of computational power of the encode process [15]. So research on efficient and fast motion estimation algorithms is significant. Block-matching algorithms (BMAs) are used widely because they are simple and easy to be applied. In the last two decades, many block-matching algorithms are proposed for alleviating the heavy computations consumed by the brute-force full search algorithm (FS) which has the best prediction accuracy, such as the new three-step search (NTSS) [16], the four-step search (4SS) [17], the block-based gradient descent search (BBGDS) [18], the diamond search (DS) [19], and the cross-diamond search (CDS) [20].

In real-world video sequences, more than 80% of the blocks can be regarded as stationary (MV = (0,0)) or quasistationary (MV = ($\pm$1,0) or (0,$\pm$1)) blocks and most of the motion vectors are enclosed in the central 5×5 pixels area for search window. TSS, NTSS, and BBGDS employ rectangular search patterns of different sizes to fit the center-biased motion vector distribution characteristics [21, 22]. Hexagon-based search employs an hexagon-shaped pattern and results in fewer search points with similar distortion [23, 24]. In this paper, a novel fast block-matching algorithm called New Cross-Hexagon Search (NHEXS) algorithm is proposed [14]. It uses small cross-shaped search patterns in the first two steps before the hexagon-based search and the proposed halfway stop technique [13]. It results in higher motion estimation speed on searching stationary and quasistationary blocks. The traditional algorithms use all the pixels of the block to calculate the distortions that result in heavy computations. We propose to use the modified partial distortion criterion (MPDC) [25] that uses certain pixels of the block, which alleviates the computations and has similar distortion.

The rest of the paper is organized as follows. The New Cross-Hexagon Search (NHEXS) algorithm is described in Section 2. The proposed improving methods for fractal video sequence coding are presented in Section 3. The experimental results are presented in Section 4. Finally the conclusions are outlined in Section 5.

2. New Cross-Hexagon Search Algorithm

2.1. Cross-Center-Biased Motion Vector Distribution. From observing the motion vector probabilities of different video sequences, we find that most real-world video sequences have the center-biased MV distribution characteristics. Motion-vector probability (MVP) can be concluded as follows. (1) Global optimal distribution is the square-center-biased (SCB) within $\pm$2 pixels, especially the zero motion vector (ZMV)(0,0); (2) MVP usually decreases away from ZMV; (3) optional MVs found along the vertical and horizontal directions are often more probable than the other locations with the same radius, which is regarded as cross-center-biased (CCB) MVP distributions. For example, there are about 15.71% and 7.94% of MVs found in vertical and horizontal directions with a radius of 1 pixel away from the ZMV. This probability is much higher than in the diagonal positions, which totally contribute about 2.76% at the same radius.

The results also show that the cross-center MV distribution is more dominant within this radius. For instance, 71.85% of MVs are found located in the central 2×2 pixels area, and there is about 68.98% of MVs located in the cross-center area. In the 4×4 pixels area, the total MVP is 81.75% and the cross-center probability within this area is 74.71%. Due to such a highly cross-biased distribution, the search pattern of BMA should match the cross-center shape to minimize the number of search points while maintaining a similar distortion error.

2.2. New Cross-Hexagon Search (NHEXS) Algorithm. The new cross-hexagon search consists of two patterns: cross-based and hexagon-based patterns. As the motion vectors distribution possesses cross-center-biased characteristics (74.76%) in the central 4×4 pixels area, two cross-shaped patterns, small-cross-shaped (SCSP) and large-cross-shaped (LCSP) patterns, as shown in Figure 1(a), are proposed as the first two initial steps to the hexagon-based search.

There are two different sizes of hexagon search patterns: large and small hexagon patterns. The large hexagon pattern used in this paper consists of not only the 7 check points in classic large hexagon pattern but also the two edge points (up and down), as shown in Figure 1(b). Therefore, the new large hexagon pattern consists of 9 search points which realizes a distinct search speed gain without increasing computational complexity of large hexagon search algorithm as shown in Figure 1(c).

From the simulation results on video sequences, we found that nearly 70% of the blocks can be regarded as stationary or quasistationary. By having this highly cross-biased property in most of the real-world sequences, we take the small cross-shaped patterns as the first two steps of NHEXS, which will save the number of search points for stationary or quasistationary blocks. Then, we search the

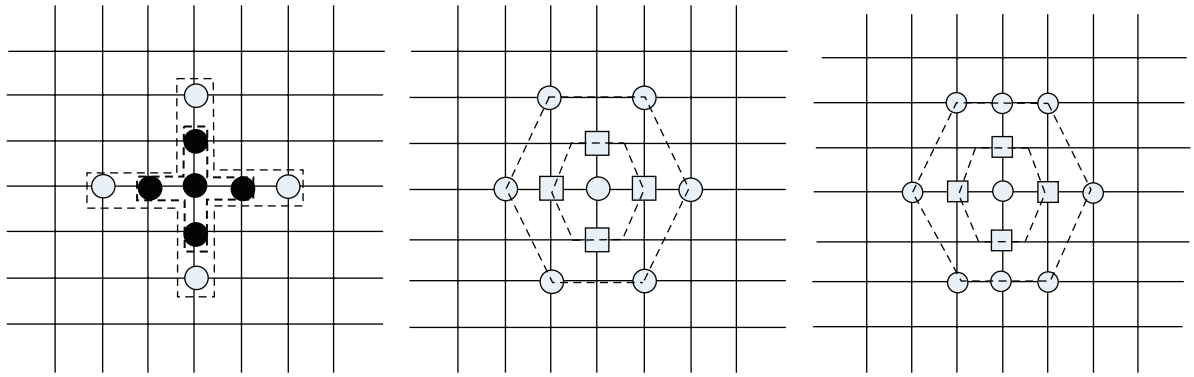

Figure 1: Search patterns used in the proposed NHEXS algorithm. (a) light gray circle: LCSP; black circle: SCSP. (b) light gray circle: classic LHEXSP; light gray square: classic SHEXSP. (c) light gray circle: LHEXSP in this paper; light gray square: SHEXSP in this paper.

remaining points of LCSP and SCB which leads to a much more precise direction for the subsequent HEXS.

The NHEXS algorithm is summarized as follows.

Step 1 (SCSP). A minimum block difference (MBD) is found from the 5 search points of the SCSP located at the center of the search window. If the MBD point occurs at the center of the SCSP, the search stops. Otherwise go to Step 2.

Step 2 (SCSP). A new SCSP is formed by using the vertex in the first SCSP as the center. If the MBD point occurs at the center of this SCSP, the search stops. Otherwise go to Step 3.

Step 3 (LCSP). The three unchecked outermost search points of the LCSP and the two unchecked points of the SCB (radius $= \pm 2$) are checked. The step tries to guide the possible correct direction for the HEXS.

Step 4 (LHEXSP). A new LHEXSP is formed by repositioning the minimum MBD found in the previous step as the center of LHEXSP. If the new MBD point is still at the center of the newly formed LHEXSP, then go to Step 5; otherwise this step is repeated again.

Step 5 (SHEXSP). Switch the search pattern from the large size of hexagon to the small size of hexagon (SHEXSP). The four points covered by the small hexagon are evaluated to compare with the current MBD point. The new MBD point is the final solution of motion vector.

A typical example is shown in Figure 2.

2.3. Analysis of NHEXS. Compared with the current HEXS [23] and cross-diamond algorithm [20], the characteristic of NHEXS algorithm lies in reducing the number of search points and increasing the search speed, especially for (quasi)stationary blocks ($|\mathrm{MV}| = 1$). For stationary blocks, HEXS and the current cross-diamond algorithm take 13 and 9 search points, respectively, while NHEXS takes 5 search points. For quasistationary blocks, HEXS and the current cross-diamond algorithm take 13 and 11 search points, respectively, while NHEXS takes 7 search points.

2.4. Modified Partial Distortion Criterion (MPDC). BMA usually uses all of the pixels in the block to calculate the distortion that causes a heavy computation [25]. In fact, we use some parts of the pixels in the block that lead to similar results.

The block size is 16×16 pixels, and the top left corner coordinates of the blocks in frame n and frame $n-1$ are (m, n) and $(m+p, n+q)$, respectively. The sum absolute difference (SAD) between the blocks in frame n and frame $n-1$ is

$$\mathrm{SAD}(m,n;p,q) = \sum_{i=0}^{15}\sum_{j=0}^{15} \left| f_n(m+i,n+j) - f_{n-1}(m+p+i,n+q+j) \right|. \tag{1}$$

$f_n(m+i, n+j)$ is the grayscale of point $(m+i, n+j)$ in frame n. $\mathrm{SAD}(m,n;p,q)$ is divided into 16 partial distortion criterions $\mathrm{sad}_k(m,n;p,q)$. The definition of the k partial distortion criterion is

$$\mathrm{sad}_k(m,n;p,q) = \sum_{i=0}^{3}\sum_{j=0}^{3} \left| \begin{array}{c} f_n(m+4i+s_k, n+j+t_k) \\ -f_{n-1}(m+p+4i+s_k, n+q+4j+t_k) \end{array} \right|. \tag{2}$$

s_k and t_k are, respectively, the horizontal and vertical distances between the top left corner point by using $\mathrm{sad}_k(m,n;p,q)$ and the top left corner of the block. The calculation of the partial distortion $\mathrm{SAD}_k(m,n;p,q)$ is defined as follows:

$$\mathrm{SAD}_k(m,n;p,q) = \sum_{j=1}^{k} \mathrm{sad}_i(m,n;p,q). \tag{3}$$

The calculation sequence of $\mathrm{sad}_k(m,n;p,q)$ $(k = 1, 2, \ldots, 16)$ is as the numbers in Figure 3, and the pixels have an equal distribution in the block. If $\mathrm{SAD}_k(m,n;p,q)$ uses too much fewer pixels, it will not replace the SAD correctly. Large simulations find that the percentage of miscarriage of justice will be below 5% if $k \geq 3$.

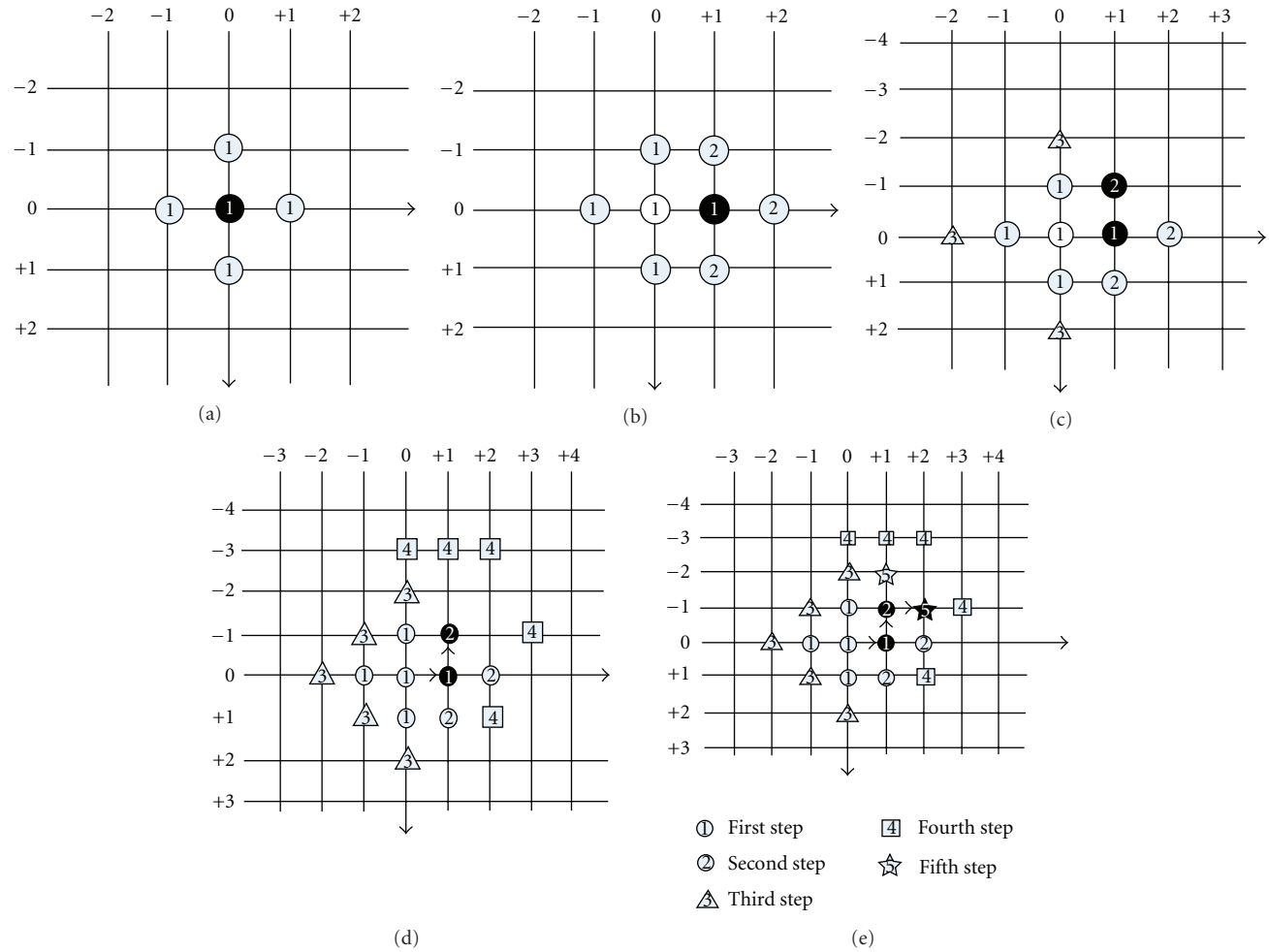

Figure 2: A search example of the NCHEXS algorithm.

3. A New Fractal Video Coding Method

3.1. Video Sequence Coding by CPM/NCIM. The CPM and NCIM combine the fractal video sequence coding with the well-known motion estimation/motion compensation (ME/MC) technique that exploits the high temporal correlations between adjacent frames. In both of the CPM and NCIM [6], each range block is motion compensated by a domain block in the previous frame, which is of the same size as the range block even though the domain block is always larger than the range block in conventional fractal coders [2, 5]. The main difference between CPM and NCIM is that CPM should be contractive for the iterative decoding process to converge, while NCIM should not be contractive since the decoding depends on the already decoded frame and is noniterative. The first two or more frames of the video sequence are treated as a coding group and are encoded by applying CPM, each frame is predicted blockwise from the n-circularly previous frame. In other words, the kth frame F_k is partitioned into range blocks, and each range block R in F_k is approximated by a domain block D in $F_{[k-1]n}$, where $[k-1]n$ denotes $k-1$ modulo n. The remaining frames are encoded by employing NCIM. The structure of NCIM is the same as that of an interframe mapping, which forms CPM, except that there is no constraint on the contrast scaling coefficients o. Since the moving image sequence exhibits high temporal correlations, this domain-range mapping becomes more effective if the size of the domain block is the same as that of the range block. In such case, the domain-range mapping can be interpreted as a kind of motion compensation. In this context, the main advantage of the proposed domain-range mapping is that, in real moving image sequence, small motion vectors are more probable than larger ones. Therefore, the search region for the motion vectors can be localized in the area near the location of the range block. In the decoder, the first n frames are reconstructed by applying CPM iteratively. Then, the remaining frames can be reconstructed by applying NCIM to the previous reconstructed frame without requiring iteration as shown in Figure 4, because the NCIM is not a contractive mapping. The first n frames encoded by CPM are the minimal decodable set of all the frames [7], in that they can be decoded without reference to other frames. Therefore, only CPM affects the convergence of the total

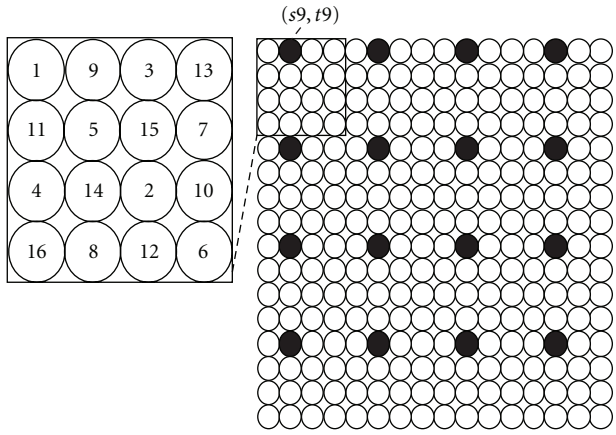

Figure 3: The calculation sequence of $\mathrm{sad}_k(m, n; p, q)$.

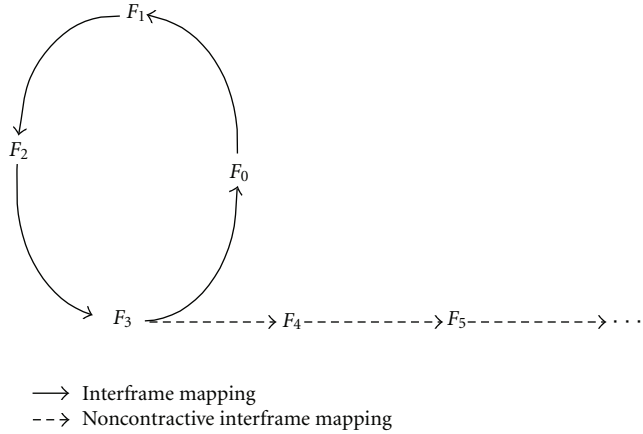

Figure 4: Hybrid structure of CPM and NCIM [6].

Table 1: The percentages of 4 modes in two video sequences.

	Mode 1	Modes 2, 3	Mode 4
"foreman.cif"	59.2%	16.3%	24.5%
"news.cif"	50.3%	28.6%	21.1%

fractal mapping and that is the reason why NCIM need not be contractive.

3.2. Using Homo-I-Frame in H.264. The original reference frame (homo-I-frame in H.264 [26]) makes a great impact on compression ratio and decoding image quality. In CPM/NCIM, the original reference frames are coded by using CPM as shown in Figure 5, and the original reference frames could be several frames which are set to four in this text.

But in CPM, the coding process involves complex block classifying, block overturning, and iteration in order to make decoding frames converge to original frames, so the compression performances are under the requirements. Then, the method based on DCT which has worked effectively in JPEG image compression standard is used to treat the original reference frame in this paper [8].

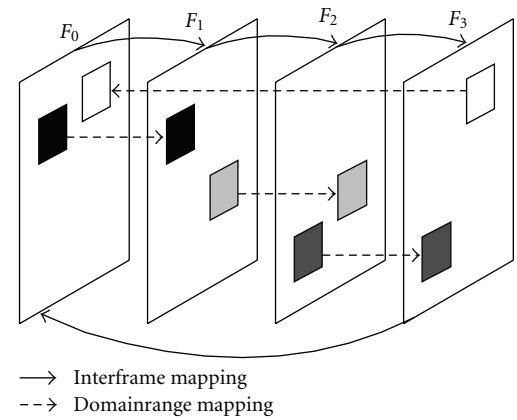

Figure 5: Circular prediction mapping.

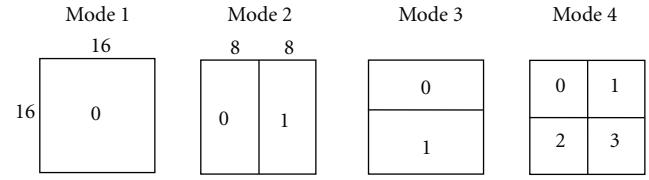

Figure 6: Macroblock partition modes.

3.3. Macroblock Partition. Macroblock partition has a large impact on calculation speed and complexity of video compression algorithm. In CPM/NCIM, a frame is partitioned by quadtree-based partition and the iteration is used in matching process which results in a high calculation complexity [6]. In this paper, a macroblock partition scheme like in H.264 is used. A frame is partitioned into many fixed-size (generally 16×16 pixels) macroblocks, and then each macroblock is partitioned as shown in Figure 6.

The matching rule in fractal image coding is the RMS:

$$\mathrm{RMS} = \frac{1}{N}\left[\sum_{i=1}^{N} r_i^2 + s\left(s\sum_{i=1}^{N} d_i^2 - 2\sum_{i=1}^{N} r_i d_i + 2o\sum_{i=1}^{N} d_i^2\right) + o\left(N.o - 2\sum_{i=1}^{N} r_i\right)\right], \tag{4}$$

$$s = \frac{N\sum_{i=1}^{N} r_i d_i - \sum_{i=1}^{N} r_i \sum_{i=1}^{N} d_i}{N\sum_{i=1}^{N} d_i^2 - \left(\sum_{i=1}^{N} d_i\right)^2}, \qquad O = \frac{1}{N}\left[\sum_{i=1}^{N} r_i - s\sum_{i=1}^{N} d_i\right]. \tag{5}$$

r_i is the pixel value of range block (R), d_i is the pixel value of domain block (D), N is the number of pixels in the block, s is the scale factor, and o is the offset factor.

The steps of macroblock partition are given as follows.

(1) Match each block (whose size is equal to current encoding block) with current macroblock and calculate the RMS in mode 1. If the minimum RMS is less than the defined threshold γ, save the IFS parameters

Table 2: The average search time per frame by total distortion criterion and modified partial distortion criterion (MPDC).

	Average search time (sec) per Frame									
	Total distortion criterion					Modified partial distortion criterion				
	NTSS	DS	CDS	HEXS	NHEXS	NTSS	DS	CDS	HEXS	NHEXS
Claire.cif	62	50	43	47	38	43	32	27	29	22
Hall.cif	56	49	42	44	36	43	31	25	29	29
Foreman.cif	65	52	45	48	40	43	34	33	33	33
News.cif	68	57	49	52	45	34	29	24	31	19
Paris.cif	69	55	50	53	46	39	35	29	29	24
Stefan.cif	72	66	57	61	50	46	39	36	35	32

Table 3: Average search points per frame.

	Search points					Comparison			
	NTSS	DS	CDS	HEXS	NHEXS	ΔNTSS	ΔDS	ΔCDS	ΔHEXS
Claire.cif	5366	3934	2813	3255	2596	−51.62%	−34.01%	−7.71%	−20.25%
Hall.cif	5673	4169	3065	3412	1795	−68.36%	−56.94%	−41.44%	−47.39%
Foreman.cif	6587	5312	4058	4513	3103	−52.89%	−41.59%	−23.53%	−31.24%
News.cif	6018	4672	3487	3796	1945	−67.68%	−58.37%	−44.22%	−48.76%
Paris.cif	6195	4980	3350	4200	3500	−43.50%	−29.72%	4.48%	−16.67%
Stefan.cif	7563	6012	5587	5677	4962	−34.39%	−17.47%	−11.19%	−12.59%

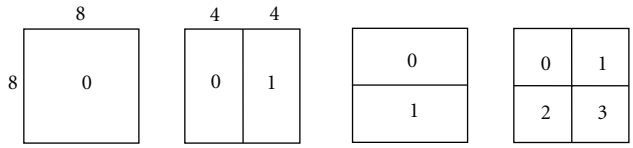

Figure 7: Subblock partition modes.

(s, o, and the domain block position "motion vector") and encode the next macroblock. Otherwise go to the next step.

(2) The RMS is calculated in mode 2 or 3. If RMS is less than γ in mode 2 or 3, then save the IFS. Otherwise go to the next step.

(3) Firstly calculate the RMS in mode 4. If the RMS is less γ, save the IFS. Otherwise each subblock in mode 4 can be partitioned continually until finding the matching block as shown in Figure 7.

The percentages of the 4 modes in two video sequences are shown in Table 1.

3.4. Encoding of IFS Parameters. The following parameters of each range block need to be encoded for subsequent transmission or storage: the scale factor s and offset intensity o; translation vector (expressed in terms of the relative position of D with respect to R) and affine transformation reduced to four rotations and four mirror reflections. Since s and o have, in general, nonuniform distributions, entropy coding usually proves to be beneficial.

We chose 5-bit quantization of s and 7-bit quantization of o, reported in the literature to give good performance [1, 9], followed by Huffman coding. To ensure that both the encoder and the decoder use the same s and o values, we quantize both during the minimization of the dissimilarity measure (4), that is, prior to each evaluation of the RMS. We encode the motion vector as a relative position of D with respect to R using fixed-length codewords (determined by the search area ±7 pixels in both horizontal and vertical directions). We use three-bit codewords for the eight possible rotations/reflections.

4. Experimental Results

4.1. The New Cross-Hexagon Search Algorithm. The proposed NHEXS algorithm is simulated using the popular video sequences (the first 70 frames of each sequence) with the size of 352×288 pixels. The macroblock is 16×16 pixels, and the maximum displacement in the search areas is ±7 pixels in both horizontal and vertical directions. The experiment is proceeded in a PC (CPU: Intel Core 2 Duo E6300, 1.86 GHz, RAM: 2G, DDR2). The algorithm is implemented in Visual C++ 6.0.

The average search times per frame by the total distortion criterion and by the modified partial distortion criterion (MPDC) are summarized in Table 2 for each sequence. The Average PSNR values and the search point numbers are summarized in Tables 3 and 4 for different algorithms including NTSS, DS, CDS, HEXS, and NHEXS.

Table 3 shows that the proposed NHEXS algorithm always consumes the smallest number of search points compared to other fast BMA. The average search points per block with the observations are NHEXS<CDS<HEXS<DS<NTSS. Especially, for sequences with motion vectors limited within a small region around (0,0) and background of small changes, such as Claire and Hall, NHEXS algorithm can

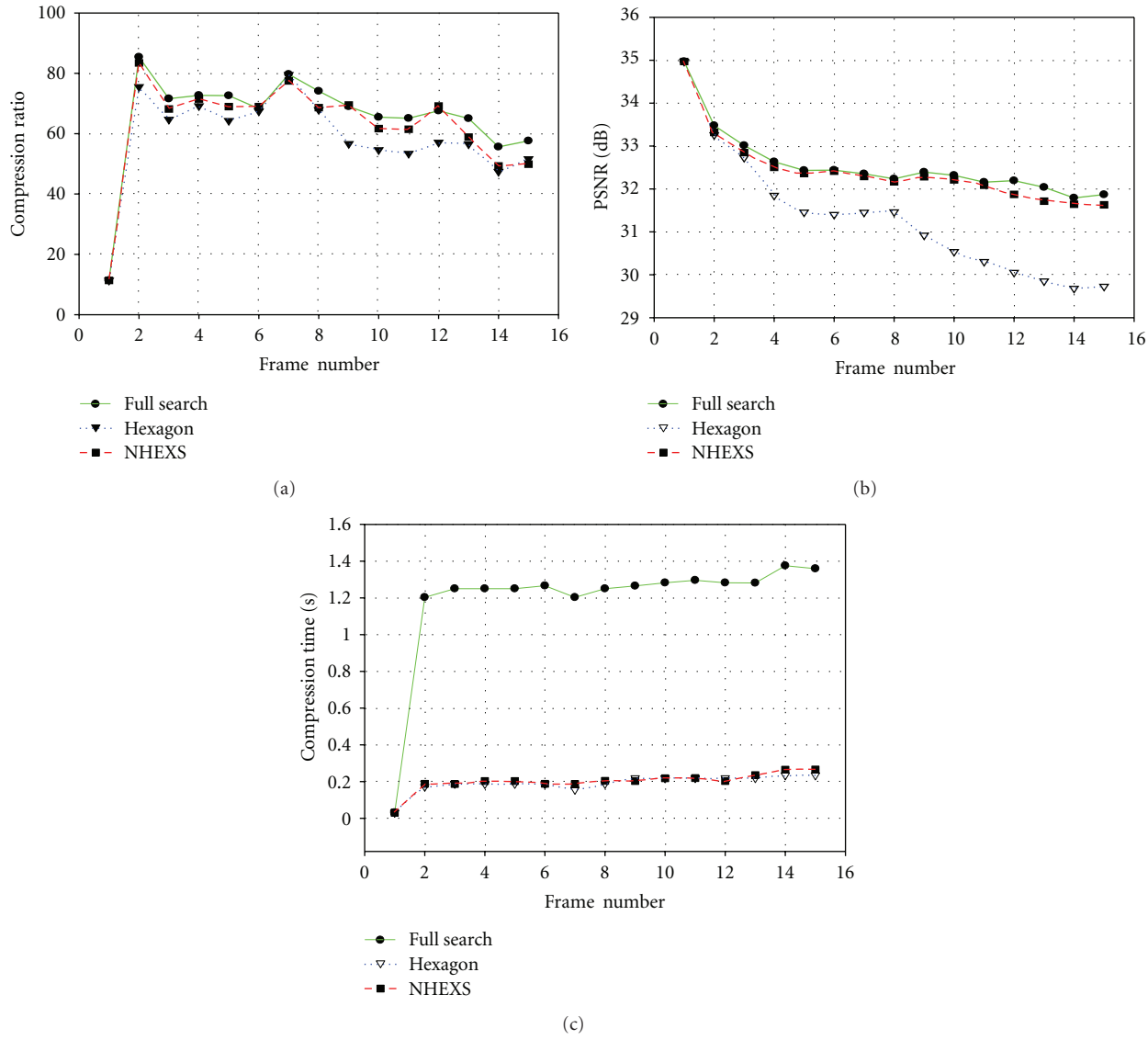

Figure 8: Comparison of full search method, hexagon method, and the NHEXS method. (a) Comparison of compression ratio. (b) Comparison of PSNR. (c) Comparison of compression time.

Table 4: Average PSNR values.

	Average PSNR Y					Comparison			
	NTSS	DS	CDS	HEXS	NHEXS	ΔNTSS	ΔDS	ΔCDS	ΔHEXS
Claire.cif	43.66	43.50	43.19	43.38	42.93	−0.73	−0.57	−0.26	−0.45
Hall.cif	40.05	40.19	39.88	39.90	39.65	−0.4	−0.54	−0.23	−0.25
Foreman.cif	35.33	35.12	35.02	35.10	34.71	−0.62	−0.41	−0.31	−0.39
News.cif	37.30	37.28	36.47	36.85	36.38	−0.92	−0.9	−0.09	−0.47
Paris.cif	33.64	33.55	36.27	36.41	36.22	2.58	2.67	−0.05	−0.19
Stefan.cif	29.23	29.01	28.57	28.74	28.20	−1.03	−0.81	−0.37	−0.54

save 45% and 28% search points than HEXS and CDS, respectively.

Table 4 shows that the average PSNR values of NHEXS have a very small decline compared with CDS and HEXS (about 0~1.4%); for sequences with background of small changes, NHEXS has almost the same average PSNR value as CDS and HEXS.

4.2. The New Fractal Video Coding. From the above we can conclude that NHEXS is robust to all kinds of sequences,

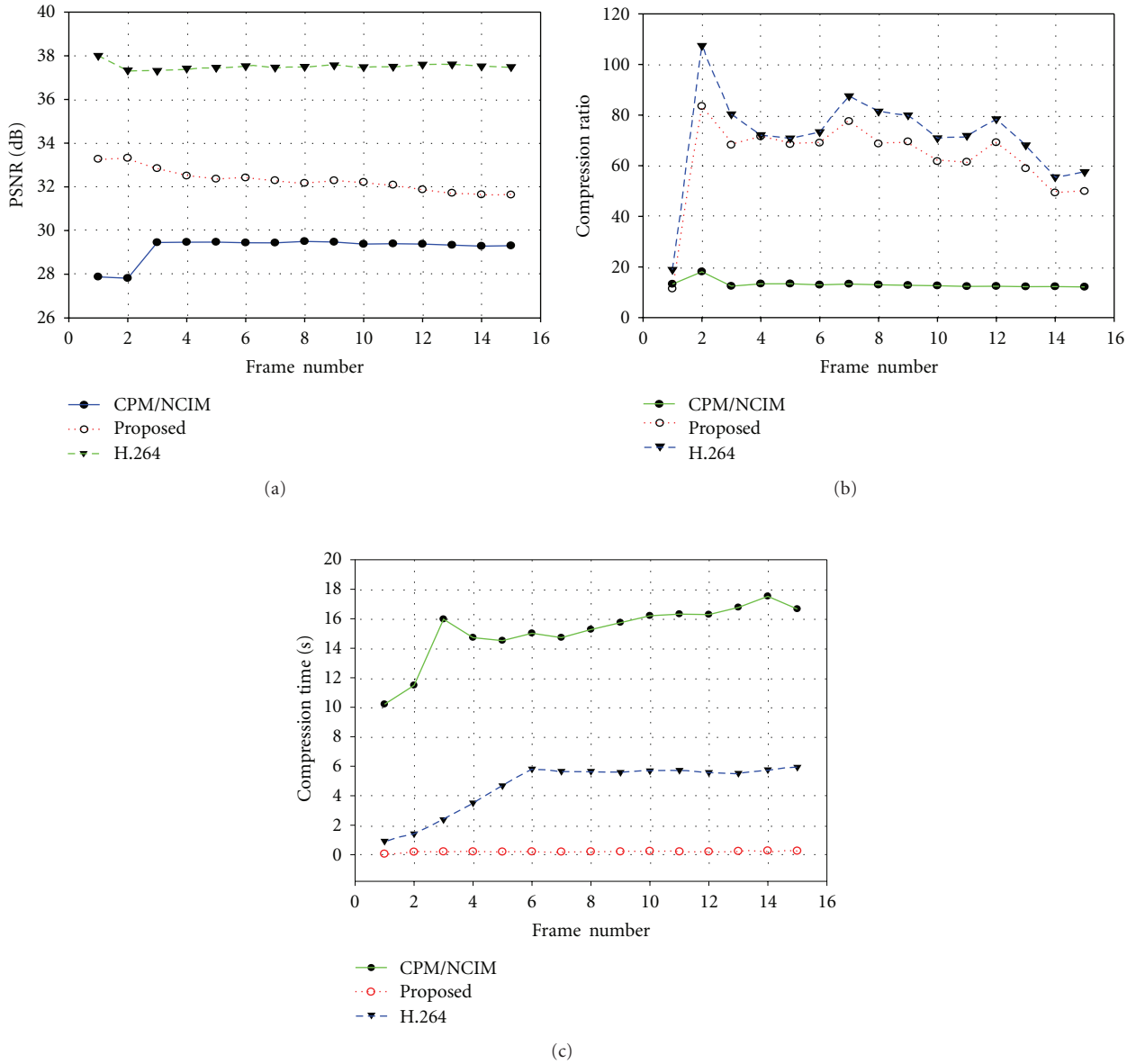

Figure 9: Comparison of the compression performance using CPM/NCIM, H.264, and the proposed method. (a) Comparison of PSNR. (b) Compression ratio. (c) Compression time.

Table 5: H264/AVC encoding parameters.

Reference software	JM14.2
Frames	15
Resolution	CIF: 352×288
GOP size	16
GOP Structure	IPPPP...
Quantization parameter	28
Search range	16
Macroblock partitioning for motion estimation	Enabled
Motion estimation method	UMHexagonS
Entropy coding method	CAVLC

which can save search time effectively and produce almost the same PSNR values compared with the popular fast BMA.

To evaluate the coding performance of the proposed algorithm, a simulation has been conducted to compare the performance of different encoding algorithms. We use 15 frames of the sequence in the simulation. The maximum and minimum partition block sizes are 16×16 pixels and 4×4 pixels, respectively. Search time and the PSNR value per frame for a sequence with medium motion, such as "foreman" (352×288 pixels, the first 15 frames), are plotted in Figure 8.

Figure 8 shows that the NHEXS method proposed in this paper makes the encoding speed 5.3 times faster than the full

Table 6: The comparison of average coding results of five video sequences.

	PSNR (dB)			Compression ratio			Time consuming (Sec)		
	CPM/NCIM	Proposed	H.264	CPM/NCIM	Proposed	H.264	CPM/ NCIM	Proposed	H.264
Foreman.cif	29.10	34.56	36.22	13.72	55.83	63.65	8.6	0.87	1.33
News.cif	31.44	35.32	36.98	20.85	91.58	112.95	6.3	0.70	1.25
Paris.cif	31.21	34.08	35.51	10.58	35.36	43.66	8.9	0.92	1.35
Bus.cif	27.71	31.56	34.72	8.39	20.69	25.90	11.18	1.23	1.78
Bridge-far.cif	30.87	35.65	37.23	30.62	159.41	202.54	6.0	0.65	1.31

search method while the quality of the decoded video images is almost the same. However the hexagon method descends a lot in PSNR.

In order to exhibit the successes of our method, the traditional CPM/NCIM method in which the CPM frames are set for 4 is also implemented.

The sequence was also compressed using the H.264/AVC reference software, version JM14.2, available for download at [26]. The sequence was encoded using the baseline profile to enable P-pictures and context adaptive variable length coding (CAVLC) for coding efficiency. Each frame was divided into a fixed number of slices, where each slice consisted of a full row of macroblocks. A fixed quantization parameter (QP = 28) was carefully selected for the sequence so as to ensure high visual quality. The sequence was visually inspected in order to check whether the chosen QPs minimized the blocking artifacts induced by lossy coding. Table 5 illustrates the parameters used to generate the compressed bitstreams.

The comparison of average coding results of five video sequences is shown in Table 6. The results indicate that the proposed scheme can raise compression ratio 4 times, speed up compression time 10 times, and improve the image quality 3 to 5 dB in comparison with CPM/NCIM. Also, the PSNR and compression ratios are low but near to H.264, and the compression speed is better than that of H.264.

The comparison is shown in Figure 9 for fifteen frames of "foreman.cif". As the frame number increases, the PSNR or the quality of decoded image decreases due to cumulating error. It could be resolved by inserting the I-frame like in H.264.

5. Conclusions

In this paper, firstly, the novel CCB characteristics of the MV distribution are exploited. With the CCB behavior in most of the real-world video sequences, we develop a novel fast algorithm using a cross-hexagon search pattern in block motion estimation, which demonstrates a significant speedup gain over the diamond-based search and other fast search methods while maintaining similar distortion performance. Meanwhile, distortion criterion is optimized, which simplifies the computational complexity without deteriorating the block distortion measure greatly.

Secondly, a fast fractal video coding method using this new cross-hexagon block-matching motion estimation is presented to reduce the encoding time and improve the encoding quality. In comparison to the CPM/NCIM method, the proposed algorithm spends less encoding time and achieves higher compression ratio and better quality compression.

References

[1] Y. Fisher, "Fractal encoding with quadtrees," in *Fractal Image Compression: Theory and Applications to Digital Images*, Y. Fisher, Ed., pp. 55–77, Springer, New York, NY, USA, 1995.

[2] Y. Fisher, T. P. Shen, and D. Rogovin, "Fractal (self-VQ) encoding of video sequences," in *Visual Communications and Image Processing*, vol. 2308 of *Proceedings of SPIE*, pp. 1359–1370, September 1994.

[3] T. Ochotta and D. Saupe, "Edge-based partition coding for fractal image compression," *The Arabian Journal for Science and Engineering*, vol. 29, no. 2, 2004, Special Issue on Fractal and Wavelet Methods.

[4] K. Belloulata, "Fast fractal coding of subbands using a non-iterative block clustering," *Journal of Visual Communication and Image Representation*, vol. 16, no. 1, pp. 55–67, 2005.

[5] Z. Yao and R. Wilson, "Hybrid 3D fractal coding with neighbourhood vector quantisation," *EURASIP Journal on Applied Signal Processing*, vol. 2004, no. 16, pp. 2571–2579, 2004.

[6] C. S. Kim, R. C. Kim, and S. U. Lee, "Fractal coding of video sequence using circular prediction mapping and noncontractive interframe mapping," *IEEE Transactions on Image Processing*, vol. 7, no. 4, pp. 601–605, 1998.

[7] J. Domaszewicz and V. A. Vaishampayan, "Graph-theoretical analysis of the fractal transform," in *Proceedings of the 20th International Conference on Acoustics, Speech, and Signal Processing (ICASSP '95)*, vol. 4, pp. 2559–2562, May 1995.

[8] M. Wang, R. Liu, and C. H. Lai, "Adaptive partition and hybrid method in fractal video compression," *Computers and Mathematics with Applications*, vol. 51, no. 11, pp. 1715–1726, 2006.

[9] K. Belloulata and J. Konrad, "Fractal image compression with region-based functionality," *IEEE Transactions on Image Processing*, vol. 11, no. 4, pp. 351–362, 2002.

[10] K. Belloulata and S. Zhu, "A new object-based system for fractal video sequences compression," in *Proceedings of the Data Compression Conference (DCC '08)*, pp. 508–510, Snowbird, Utah, USA, March 2008.

[11] K. Belloulata and S. Zhu, "A new object-based fractal stereo codec with quadtree-based disparity or motion compensation," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '06)*, vol. 2, pp. 481–484, Toulouse, France, May 2006.

[12] S. Zhu and K. Belloulata, "A novel object-based fractal stereo video codec," in *Proceedings of the IEEE International Conference on Image Processing (ICIP '05)*, vol. 1, pp. 805–808, Genova, Italy, September 2005.

[13] S. Zhu, J. Tian, X. Shen, and K. Belloulata, "A new cross-diamond search algorithm for fast block motion estimation," in *Proceedings of the IEEE International Conference on Image Processing (ICIP '09)*, vol. 1, pp. 1581–1584, Cairo, Egypt, November 2009.

[14] S. Zhu, J. Tian, X. Shen, and K. Belloulata, "A novel cross-hexagon search algorithm based on motion vector field prediction," in *Proceedings of the IEEE International Symposium on Industrial Electronics, (ISIE '09)*, pp. 1870–1874, Seoul, Korea, July 2009.

[15] "Information technology-coding of audio visual objects-part 2: visual," ISO/IEC 14 469-2 (MPEG-4 Visual), 1999.

[16] R. Li, B. Zeng, and M. L. Liou, "New three-step search algorithm for block motion estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 4, no. 4, pp. 438–442, 1994.

[17] L. M. Po and W. C. Ma, "A novel four-step search algorithm for fast block motion estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, no. 3, pp. 313–317, 1996.

[18] L. K. Liu and E. Feig, "A block-based gradient descent search algorithm for block motion estimation in video coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, no. 4, pp. 419–422, 1996.

[19] S. Zhu and K. K. Ma, "A new diamond search algorithm for fast block-matching motion estimation," *IEEE Transactions on Image Processing*, vol. 9, no. 2, pp. 287–290, 2000.

[20] C. H. Cheung and L. M. Po, "A novel cross-diamond search algorithm for fast block motion estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 12, pp. 1168–1177, 2002.

[21] J. Y. Tham, S. Ranganath, M. Ranganath, and A. A. Kassim, "A novel unrestricted center-biased diamond search algorithm for block motion estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, no. 4, pp. 369–377, 1998.

[22] H. M. Chen, P. H. Chen, K. L. Yeh, W. H. Fang, M. C. Shie, and F. Lai, "Center of mass-based adaptive fast block motion estimation," *EURASIP Journal on Image and Video Processing*, Article ID 65242, 11 pages, 2007.

[23] C. Zhu, X. Lin, and L. P. Chau, "Hexagon-based search pattern for fast block motion estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 5, pp. 349–355, 2002.

[24] A. Hamosfakidis and Y. Paker, "A novel hexagonal search algorithm for fast block matching motion estimation," *Eurasip Journal on Applied Signal Processing*, vol. 2002, no. 6, pp. 595–600, 2002.

[25] C. K. Cheung and L. M. Po, "Normalized partial distortion search algorithm for block motion estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 10, no. 3, pp. 417–422, 2000.

[26] Joint Video Team (JVT), "H.264/AVC reference software version JM14.2," http://iphome.hhi.de/suehring/tml/download/.