Received August 5, 2017, accepted August 17, 2017, date of publication September 1, 2017, date of current version October 12, 2017.

Digital Object Identifier 10.1109/ACCESS.2017.2745538

An Improved Inter-Frame Prediction Algorithm for Video Coding Based on Fractal and H.264

SHIPING ZHU, (Member, IEEE), SHUPEI ZHANG, AND CHENHAO RAN
Department of Measurement Control and Information Technology, School of Instrumentation Science and Optoelectronics Engineering, Beihang University, Beijing 100191, China

Corresponding author: Shiping Zhu (spzhu@163.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61375025, Grant 61075011, and Grant 60675018 and in part by the Scientific Research Foundation for the Returned Overseas Chinese Scholars from the State Education Ministry of China.

ABSTRACT Video compression has become more and more important nowadays along with the increasing application of video sequences and rapidly growing resolution of them. H.264 is a widely applied video coding standard for academic and commercial purposes. And fractal theory is one of the most active branches in modern mathematics, which has shown a great potential in compression. In this paper, this study proposes an improved inter prediction algorithm for video coding based on fractal theory and H.264. This study take the same approach to make intra predictions as H.264 and this study adopt the fractal theory to make inter predictions. Some improvements are introduced in this algorithm. First, luminance and chrominance components are coded separately and the partitions are no longer associated as in H.264. Second, the partition mode for chrominance components has been changed and the block size now rages from 16×16 to 4×4, which is the same as luminance components. Third, this study introduced adaptive quantization parameter offset, changing the offset for every frame in the quantization process to acquire better reconstructed image. Comparison between the improved algorithm, the original fractal compress algorithm and JM19.0 (The latest H.264/AVC reference software) confirms a slightly increase in Peak Signal-to-Noise Ratio, a significant decrease in bitrate while the time consumed for compression remains less than 60% of that using JM19.0.

INDEX TERMS Video compression, fractal theory, H.264/AVC, inter-frame prediction.

I. INTRODUCTION

H.264 is a digital video coding standard developed by JVT, and it was officially released in March, 2003 [1], [2]. H.264 relies on 4×4 integral discrete cosine transform (DCT) and variable-length code (VLC), and it is more effective in motion estimation than other standards. The bitrate of H.264 coded videos is merely half of that of H.263, given the same quality of video [3]. Due to its remarkable compression efficiency and adaptability to network conditions, H.264 is widely used for multimedia applications. However, H.264 is too computationally intensive and time consuming to be directly applied on real-time systems. According to Acharjee *et al.* [4], motion estimation involves the most computation as compared to other processes in the entire video compression process. Hence, the coding time can be greatly shortened by reducing the computational complexity of motion estimation.

Fractal image coding provides a scheme to code image based on the similarity between different parts of an image or the similarity between different images. It features high compression ratio, short coding time but the quality of coded image is not satisfying [5]–[8]. Beaumont pioneered to apply Jacquin's image compression theory on video compression. Although the compression ratio was high, the quality of the image was poor and the blocking effect was obvious [9]. Zhu *et al.* [10] combined the object-based video coding and fractal coding algorithms, dividing each frame of the video into foreground and background for separate compression. After combining the advantages of several methods, Farkade *et al.* [11] proposed a fast-search fractal algorithm based on block matching to reduce time consumption for block matching. Chaudhari and Dhok proposed a novel full-search algorithm for motion estimation [12]. In their method, cross-correlated cyclic FFT is used as the condition for full-search block matching. In addition to performance gains in terms of compression ratio and quality, the speed was almost 10 times faster than the traditional full-search compression scheme. Kamble *et al.* used a modified three-step search algorithm for block-matching motion estimation to reduce computational cost [13]. And some recent studies focus on reduce coding time by propose new algorithm involving quad-tree scheme et al [14]–[18], or utilizing more powerful hardware like GPUs, FPGAs or cloud computing [17]–[20].

H.264 and fractal image coding both have their own pros and cons. H.264 is computationally intensive but the quality of the image is great. The reverse is true for fractal image coding. Hence, striking a balance between image qualities and coding speed by using H.264 for intra prediction and using fractal coding for inter prediction may produce some satisfying results. Based on this idea, this study proposes a new video coding algorithm to improve image quality and reduce bitrate.

The rest of this paper is organized as follows. Section II focuses on the intra coding process of H.264. Section III describes the mathematical fundamentals of fractal coding and the detailed process of the fractal coding algorithm adopted in this paper. Section IV discusses improvements made to the fractal coding algorithm. Section V compares the performance of the original fractal coding algorithm, the improved algorithm and JM19.0 [21], [22], the latest reference software of H.264/AVC.

II. OVERVIEW OF INTRA CODING IN H.264/AVC

Three types of frames are defined in H.264. They are the I frame, the P frame and the B frame.

The frame that makes prediction based on the current frame itself is called the I frame. The I frame can be decoded independently without the use of a reference. The first frame of a video sequence is always an I-frame, and its defect lies in the need to occupy a large number of bits.

The P frame is coded based on previous I or P frames. It usually consumes fewer bits than I frame, but it is prone to error during transmission, due to its heavy dependence on the previous reference frame.

The B frame jointly refers the previous and subsequent frames. Its difference with the P frame is that the B frame can refer both the previous and following I and P frames.

The coding process of the I frame is called the intra-frame coding. Macroblock in the luminance component is either 16×16 or 4×4 for intra-frame prediction. The size of macroblock is only 8×8 for the chrominance component [21], [22].

A. 4×4 LUMINANCE PREDICTION MODE

In the case of macroblock that has complicated contents, the error produced by predicting via 4×4 prediction is smaller than that via 16×16 prediction. During the prediction, 9 prediction modes are traversed and their rate-distortion costs are computed, respectively. The mode with the lowest cost is regarded as the final mode for prediction. The 9 prediction modes are shown in Fig. 1.

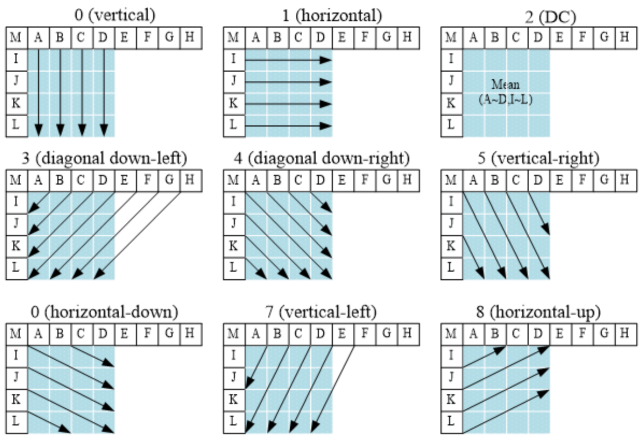

FIGURE 1. Prediction mode for 4×4 luminance components.

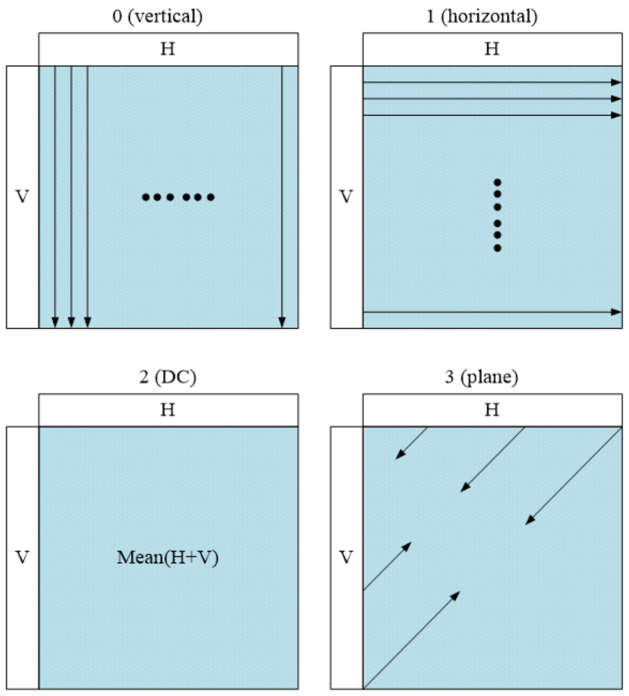

FIGURE 2. Prediction mode for 16×16 luminance components.

B. 16×16 LUMINANCE PREDICTION MODE AND 8×8 CHROMINANCE PREDICTION MODE

Adopting the 16×16 prediction mode for the luminance macroblock that has few contents can reduce bitrate while guaranteeing relatively small errors. And the 8×8 prediction mode is adopted by all chrominance macroblocks. Four prediction modes need to be traversed in those two macroblock prediction modes. The one with the smallest value of RDcost will be regarded as the final mode. The four prediction modes are shown in Fig. 2.

H.264 provides a broad variety of prediction modes, and it is thus of great importance to determine how to choose the optimal mode. The Lagrange strategy for rate-distortion optimization (RDO) is used in H.264 to choose the optimal coding mode:

1) INTRA 4×4 PREDICTION

In the case of intra-frame 4×4 prediction, the cost of coding with one specific mode is calculated using (1), and the mode with the minimal cost is the optimal mode.

$$J\left(s, c, IMODE|QP, \lambda_{MODE}\right) = SSD\left(s, c, IMODE|QP\right) + \lambda_{MODE} R\left(s, c, IMODE|QP\right) \tag{1}$$

Where c denotes the macroblock reconstructed after prediction, DCT, quantization, IDCT and inverse quantization; s denotes the original macroblock; SSD denotes the square sum of the difference between s and c; R denotes the number of bits needed to code the macroblock with IMODE and quantization parameter(QP); IMODE denotes the 9 intra prediction modes in Fig. 1.

2) INTRA 16×16 PREDICTION

In the case of intra-frame 16×16 prediction, the mode that can minimize SATD of s and c is chosen, where SATD denotes the transformed sum of errors. It can indicate the level of both distortion and the bitrate.

$$J\left(s, c, IMODE|QP, \lambda_{MODE}\right) = SATD\left(s, c, IMODE|QP\right) \tag{2}$$

Because the predicted value of the pixel is close to the original value, it suffices to only transmit the error between them. And it can be computed as (3) shows:

$$e_n = f_n - f_n' \tag{3}$$

The residuals are mostly small, thus the original data are compressed.

III. MATHEMATICAL BACKGROUND OF FRACTAL COMPRESSION AND FRACTAL VIDEO COMPRESSION ALGORITHM

A. MATHEMATICAL BACKGROUND OF FRACTAL COMPRESSION

1) COMPLETE METRIC SPACE

A metric space (X, d) consists a space or an non-empty set X and a real-valued function $d : X \times X \to R$. For any $x, y, z \in X$, it has the following properties:

- $d\left(x, y\right) \geq 0$;
- $d\left(x, y\right) = 0$, *if and only if* $x = y$;
- $d\left(x, y\right) = d\left(y, x\right)$;
- $d\left(x + y\right) + d\left(y, z\right) \geq d\left(x, z\right)$.

In these Equations, d is the metric defined in this space, and it is also called distance function or simply distance. Generally, if it can be inferred from the context that the metric is used, then d is usually omitted and X is regarded as the metric space.

If all Cauchy sequences in the metric space (X, d) are convergent, or if each Cauchy sequence converges to X, then (X, d) is called a complete metric space or Cauchy space.

2) CONTRACTION MAPPING

Let (X, d) denote the metric space, and w denote the mapping X→X. If there exists a positive real number s which satisfies (4) for any x, y ∈ X, then this mapping is Lipschitz continuous on s.

$$d(\omega\left(x\right), \omega\left(y\right)) \leq s \cdot d(x, y) \tag{4}$$

If the Lipschitz constant s<1, ω is called the contraction mapping on s.

3) FIXED POINT OF CONTRACTION MAPPING

If X is a complete metric space and $\omega : X \to X$ is a contraction mapping, there exists one single point $x_\omega \in X$ which satisfies (5) for any x ∈ X.

$$x_\omega = f\left(x_\omega\right) = lim_{n\to\infty} f^n(x) \tag{5}$$

This point is called the fixed point or the attractor of the mapping f. If a mapping brings points close to each other, this mapping is contracted. For example, if the mapping f (x) = x/2 is contracted, given a random original value x, the computed results of f(x), f(f(x)), $f^3(x)$, ... will converge to the fixed point 0.

4) GENERALIZED COLLAGE THEOREM

If f is convergent to the exponent n, there exists only one fixed point $x_\omega \in X$, which can satisfy (6) for any x ∈ X.

$$x_\omega = f\left(x_\omega\right) = lim_{n\to\infty} f^n(x) \tag{6}$$

In this case, we have:

$$d(x, x_\omega) \leq \frac{1}{1-s}\frac{1-\delta^n}{1-\delta} d(x, f(x)) \tag{7}$$

where s is the contraction factor of f^n, and the Lipschitz factor of f.

Suppose X is a complete metric space, and the contraction mapping $\omega_i : X \to X(i = 1, 2, \ldots, n)$ constitutes an iterative function system (IFS). From the contraction mapping theory on fixed point, it is known that IFS is the set that defines the sole attractor for the mapping. Because the attractor is unique, it is absolutely dependent on the mapping ω and its original value.

Fractal image coding is quite an opposite process of the discussion above, where an IFS that takes the full image as an attractor needs to be found. However, there is no general solution to this problem nowadays, because a normal image has local autocorrelation only. Hence, IFS is extended to PIFS (Partitioned IFS), which consists of the contraction mapping $\omega_i : D_i \to X(i = 1, \ldots, n)$, where $D_i \in X(i = 1, \ldots, n)$. Thanks to the properties of PIFS, more generic and non-autocorrelated sets can be coded through collage coding, so that each part in a region can always find other parts that resemble to it.

TABLE 1. Affine transform matrix.

T_0	T_1	T_2	T_3
$\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}$	$\begin{bmatrix}1 & 0\\0 & -1\end{bmatrix}$	$\begin{bmatrix}-1 & 0\\0 & 1\end{bmatrix}$	$\begin{bmatrix}-1 & 0\\0 & -1\end{bmatrix}$
T_4	T_5	T_6	T_7
$\begin{bmatrix}0 & 1\\1 & 0\end{bmatrix}$	$\begin{bmatrix}0 & 1\\-1 & 0\end{bmatrix}$	$\begin{bmatrix}0 & -1\\1 & 0\end{bmatrix}$	$\begin{bmatrix}0 & -1\\-1 & 0\end{bmatrix}$

5) AFFINE TRANSFORM

The general form of an affine transform for the 2-D Euclidean space is:

$$W(x) = W\begin{bmatrix}x_1\\x_2\end{bmatrix} = \begin{bmatrix}a & b\\c & d\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix} + \begin{bmatrix}e\\f\end{bmatrix} \tag{8}$$

where the matrix $\begin{bmatrix}a & b\\c & d\end{bmatrix}$ is equal to $\begin{bmatrix}r_1\cos\theta_1 & -r_2\sin\theta_2\\r_1\sin\theta_1 & r_2\cos\theta_2\end{bmatrix}$. So, (8) can be written as:

$$W(x) = W\begin{bmatrix}x_1\\x_2\end{bmatrix} = \begin{bmatrix}r_1cos\theta_1 & -r_2sin\theta_2\\r_1sin\theta_1 & r_2cos\theta_2\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix} + \begin{bmatrix}e\\f\end{bmatrix} \tag{9}$$

The affine transform can be performed to skew, extrude, rotate and converge the input image or change the scale of the input image.

B. FRACTAL VIDEO COMPRESSION

1) OVERVIEW

There might be local similarity between different partitions in a normal image. Hence, one partition can be collaged into another one through affine transform [23].

Consider an m×m grey level image. Assume that all n×n macroblocks that do not overlap one another constitute the range pool, and an image has $(m/n)^2$ macroblocks at most. In order to conform to the contraction mapping theory on fixed point, the size of the domain block is at least twice that of the range pool. Therefore, the domain pool D consists of all 2n×2n macroblocks. The macroblocks can be overlapped with one another, and thus there are at most $(m - 2n + 1)^2$ macroblocks in the domain pool. Let R denote the range pool and v denote the macroblock in range pool. The affine transform is constructed by searching D for a domain block that is the most similar to the range block being coded. Several parameters that represent the affine transform will be calculated and will be part of the fractal coding result of a macroblock.

In order to measure the correlation between R blocks and D blocks, the D blocks must be resampled to 8×8 blocks to ensure they have the same size as R blocks. The resized D blocks will be called u here. The correlation of two n×n macroblocks can be quantified by calculating their MSE. Definition of MSE is given in (10).

$$MSE = \left[\frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left(f'(x,y) - f(x,y)\right)^2\right]^{1/2} \tag{10}$$

In fractal affine transform, there are 8 transforms $T_k : k = 0, 1, \ldots, 7$ that can be applied to a domain block, as shown in Table I.

The constant scale factor p and the offset q can be taken into account in the fractal affine transform. Therefore, the fractal affine transform φ of u(x, y) can be written as:

$$\varphi\begin{bmatrix}x\\y\\u(x,y)\end{bmatrix} = \begin{bmatrix}a_{11} & a_{12} & 0\\a_{21} & a_{22} & 0\\0 & 0 & p\end{bmatrix}\begin{bmatrix}x\\y\\u(x,y)\end{bmatrix} + \begin{bmatrix}t_x\\t_y\\q\end{bmatrix} \tag{11}$$

where the matrix $\begin{bmatrix}a_{11} & a_{12}\\a_{21} & a_{22}\end{bmatrix}$ denotes one of the transform matrixes in Table 1, (t_x, t_y) denotes the coordinate of the domain block. The similarity is quantified by calculating $d = \|p \cdot u_k + q - v\|$. Then, p and q can be directly computed as:

$$p = \frac{[N\langle u_k, v\rangle - \langle u_k, \hat{1}\rangle\langle v, \hat{1}\rangle]}{[N\langle u_k, u_k\rangle - \langle u_k, \hat{1}\rangle^2]} \tag{12}$$

$$q = \frac{1}{N}[\langle v, \hat{1}\rangle - p\langle u_k, \hat{1}\rangle] \tag{13}$$

where N is the number of pixels in the range block, and $\hat{1} = [11\ldots1]^T$.

Finally, after u traverses the entire domain pool, the set of parameters (t_x, t_y), p, q and k will be obtained, which provide the information needed to perform fractal affine transform on the macroblock v. The coding process is completed after v traverses the entire range pool [24], [25].

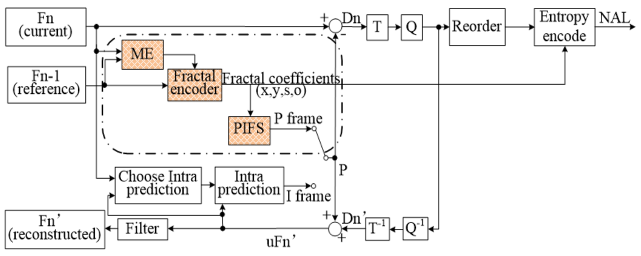

FIGURE 3. The structure of the proposed fractal coding system.

2) ALGORITHM

The structure of the fractal coding system is shown in Fig. 3.

The I frame in the proposed system adopts the same coding method for I frame in H.264 to improve the quality of the image, and the P frame adopts the fractal coding algorithm to accelerate the coding speed. In H.264, intra-frame and inter-frame coding are both available for macroblocks in a P frame. But in this paper, macroblocks in a P frame can only make inter-frame predictions. The coding process is presented in Fig. 4.

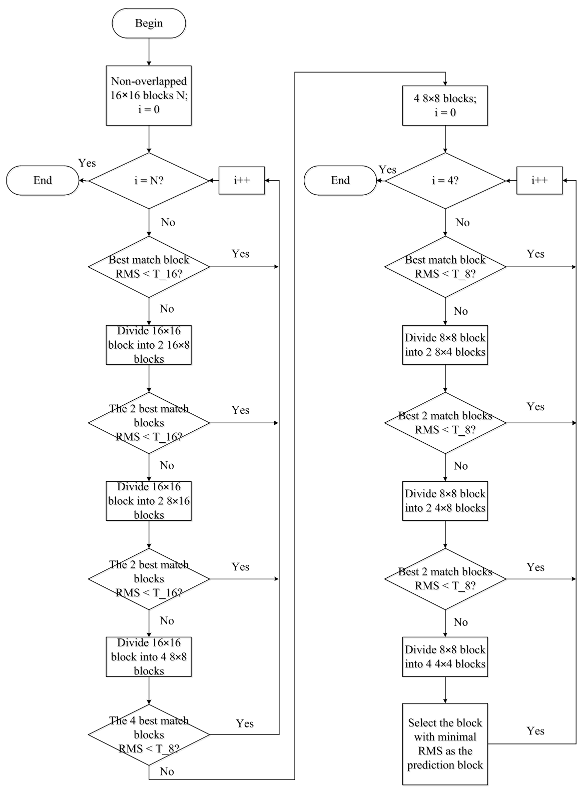
Begin
Non-overlapped 16×16 blocks N; i = 0
End
Yes
i = N?
i++
No
Best match block RMS < T_16?
Yes
No
Divide 16×16 block into 2 16×8 blocks
The 2 best match blocks RMS < T_16?
Yes
No
Divide 16×16 block into 2 8×16 blocks
The 2 best match blocks RMS < T_16?
Yes
No
Divide 16×16 block into 4 8×8 blocks
The 4 best match blocks RMS < T_8?
Yes
No
4 8×8 blocks; i = 0
End
Yes
i = 4?
i++
No
Best match block RMS < T_8?
Yes
No
Divide 8×8 block into 2 8×4 blocks
Best 2 match blocks RMS < T_8?
Yes
No
Divide 8×8 block into 2 4×8 blocks
Best 2 match blocks RMS < T_8?
Yes
No
Divide 8×8 block into 4 4×4 blocks
Select the block with minimal RMS as the prediction block
Yes

FIGURE 4. Block diagram of the proposed encoder system.

Firstly the first I frame will be coded using H.264, and use the reconstructed image as the reference frame of the subsequent P frame. As for the coding of the P frame, the range block is located in the current frame and the domain block is within the reference frame. Search D for the optimal matching block of the current block R.

Both I frame and P frame can be the reference of a P frame. The best partitioning method will be determined by calculating the sum of squared residuals between the original image and the reconstructed image.

Details of the process for P frame coding are shown in Fig. 4.

(a) If the current frame is an I frame (the first frame of each video sequence is definitely an I frame), make prediction using H.264, record the prediction mode and residuals.

(b) If the current frame is a P frame, take the previous frame as a reference frame to perform fractal coding, record the fractal parameters.

(c) Transform and quantify the residuals, then write the residuals and fractal parameters to bit stream after entropy coding.

(d) Reconstruct the frame (or macroblock) using transformed and quantified residuals and fractal parameters. In intra-frame coding, the reconstructed macroblock will be used as the reference of other macroblocks directly while in inter-frame coding the reconstructed frame will first be deblocking filtered and then used as a reference.

Fractal affine transform constructs every macroblock v in R by searching the domain block within ±7 pixels near the R macroblock. Note that the searched region can be enlarged or narrowed, and the setting of ±7 pixels is an optimal value determined in our experiment. If the minimum RMS (Root Mean Square) between the macroblock v and the domain block is less than T_16 (a threshold), then the current domain block is regarded as a best match and proceed to the next macroblock. RMS is defined as:

$$RMS = \frac{1}{N}\left[\begin{array}{l}\sum_{i=1}^{N} r_i^2 + s\left(s\sum_{i=1}^{N} d_i^2 - 2\sum_{i=1}^{N} r_i d_i + 2o\sum_{i=1}^{N} d_i\right) \\ +o(N\cdot o - 2\sum_{i=1}^{N} r_i)\end{array}\right] \tag{14}$$

where the parameters s and o are defined as:

$$s = \frac{N\sum_{i=1}^{N} r_i d_i - \sum_{i=1}^{N} r_i \sum_{i=1}^{N} d_i}{N\sum_{i=1}^{N} d_i^2 - (\sum_{i=1}^{N} d_i)^2} \tag{15}$$

$$o = \frac{1}{N}\left[\sum_{i=1}^{N} r_i - s\sum_{i=1}^{N} d_i\right] \tag{16}$$

In these equations, N denotes the number of pixels in the macroblock, r_i denotes the values of pixels in the range block, d_i denotes the values of pixels in the domain block. The process to derive the fractal coefficients s and o are available in the Appendix. Finally, the reconstructed pixel value R' is:

$$R' = s \cdot D + o \tag{17}$$

If RMS is larger than T_16, the macroblock v will be partitioned into two 16×8 sub-macroblocks, each of which will be predicted separately. And their RMS will be computed, respectively. If both of their RMS values are less than T_16, then the optimal match is found and the program will proceed to the next macroblock.

If the requirements are not satisfied, the macroblock will be partitioned into two 8×16 sub-macroblocks, compute their RMS and compare them with the threshold. If the requirements are met, the program will proceed to the next macroblock; otherwise, the macroblock v will be partitioned into four 8×8 sub-macroblocks. At this time, the threshold changes to T_8. The same process above is repeated to check whether the requirement is met. If the 8×8 block satisfies the requirement, the program will proceed to the next macroblock. Otherwise, it will be partitioned into the 8×4 blocks and repeat the process.

Throughout the process, the macroblock has 7 partition modes that ranges from 16×16 to 4×4. When one mode does not satisfy the requirement, the program will proceed to partition the macroblock downwards to the next mode until it reaches 4×4. In the case of 4×4, the match which has the minimal RMS will be regarded as the optimal match and indicates the program can proceed to the next macroblock.

A prediction frame will be generated after all macroblocks are predicted. The difference between the prediction frame and the current frame is called the residuals frame. The transformed and quantified residuals are written into the bit stream. The quantified data is then subjected to inverse quantization and inverse transform, and added to the prediction frame to form the reconstruction frame, which will be stored for prediction of subsequent frames.

IV. IMPROVEMENT TO THE ORIGINAL ALGORITHM

A. CHROMINANCE MACROBLOCK RESIZING

In H.264, the 8×8-2×2 block partitioning mode is adopted for both intra-frame and inter-frame coding for chrominance components. Because the number of pixels in the chrominance component is one fourth that of the luminance component, each 8×8 chrominance block corresponds to a 16×16 luminance block, sharing the same motion vector and macroblock partitioning. In normal video sequences, the contents of chrominance components are often closely related to that of luminance components, so they share the same motion vector and partition mode with the corresponding luminance component instead of being predicted separately. This is a highly effective method to reduce the bitrate and coding time in H.264.

While making predictions, the fractal coding scheme should not merely consider the motion vector by finding the exactly identical or most similar blocks. Instead, fractal matching between blocks, which probably involve fractal affine transforms, should also be taken into account. Hence, it is very possible that the luminance block and the chrominance block do not share the same motion vector and fractal partition mode, underscoring the need to code them separately.

The chrominance and luminance components have their own fractal parameters, motion vectors and fractal modes when they are separately coded. If the 8×8-2×2 partition mode is applied, the number of motion vectors will be at least twice that of H.264. At the same time, bitrate will continue to increase due to the fractal parameters. Therefore, the partitioning mode of the chrominance component is switched to 16×16-4×4, the same as the luminance components. Then the chrominance component can share the same coding process as the luminance component because they have exact the same size.

Adopting the new partition mode reduces the number of macroblocks of the chrominance component by 3/4. Our experimental results show that the coding time is reduced by over 40% while maintaining almost the same Peak Signal-to-Noise Ratio (PSNR) for chrominance components. Details of the experimental results are given in Section V.

B. SELF-ADAPTIVE QP OFFSET

Quantization provides an extremely effective tool to reduce the bitrate of a video. It needs a lot of bits to represent a numerical value that has a large dynamic range. But after quantization, each value can be represented with fewer bits, for the data in a certain range is mapped to a same word based on a rule. The quantization process of a uniform scalar quantizer can be written as:

$$Z = floor(|W|/\Delta) \cdot sgn(W) \tag{18}$$

where Δ denotes the quantization step length, the function floor() denotes the operation to round down a number to the nearest integer, sgn() denotes the sign function that extract the sign of a real number.

Accordingly, the inverse quantization equation is:

$$W' = \Delta \cdot Z \tag{19}$$

It can be seen that the quantization step length is an important parameter to determine the accuracy and bitrate of an image. A larger step length means that fewer bits are needed for coding a numerical value and that more information about the video is lost.

The values in the range of 0-Δ will be mapped to 0 after quantization, causing inevitable quantization errors. Hence, H.264 defines a concept of quantization offset f. After adding the offset to Equation (18), the quantization equation is:

$$Z = floor(\frac{|W| + f}{\Delta}) \cdot sgn(W) \tag{20}$$

The inverse quantization equation remains the same. After improving the quantizer, the deadband of quantization can be controlled by changing the value of f. If W is in the range $(-\Delta + f, \Delta - f)$, the quantized value is zero. The deadband of quantization decreases with f. Note that the size of the deadband influences the subjective quality of the image. If the deadband is too large, many small-value pixels will be quantized to 0, causing the loss of many minutiae of the image. Based on statistical results, H.264 defines $f = \Delta/3$ for intra-frame prediction and $f = \Delta/6$ for inter-frame prediction.

However, setting the value of f like this relies on statistical results, and it is not the optimal quantization offset for the coding of each frame. It is learned from our experiment that the quantization offset has a great influence on the PSNR of an image. Taken in to consideration the fact that the offset is not written into the bit stream, the quality of image can be optimized by adjusting the quantization offset for each frame without influencing the size of the bit stream. Motivated by this idea, this study proposes a self-adaptive QP offset algorithm. Details are as follows.

Step 1: After fractal prediction, compute the residuals and then proceed to choose the QP offset.

Step 2: Empirically pre-set the range of the QP offset, and ensure that this range can be narrowed according to statistical results. QP offset is not set up directly. Instead, it is related to the value of QP, and it can be computed as (21):

$$Q_{step} = Q(QP\%6) \cdot 2^{QP/6} \tag{21}$$

$$f = Q_{step}/denom \tag{22}$$

where *Q(QP%6)* refers to the Q_{step} that corresponds to $QP = 1 - 6$, and it can be retrieved quickly using the preset data array; denom denotes the factor used to set the quantization offset f, and it is also an important parameter to

TABLE 2. **Configuration.**

Input Benchmarks Sequences (CIF)	Low speed	foreman, bridge close, mobile
	Normal speed	coastguard, flower, paris
	High speed	bus, football, stefan
Profile IDC	Baseline	
LevelIDC	40	
Frame rate	30 Hz	
Number of reference frame in each reference list	1	
Frame encoded	1 Intra followed by 9 Inter frames	
Intra period	10	
No. of B frames	0	
Frames to be encoded	30	
QP	0,4,8,14,20,28,36,51	
Hadamard transform	Enabled	
RDO	Enabled	
Motion search range	±7 pixels with ¼-pel accuracy	
Motion estimation	Full search	
Symbol mode	CAVLC	

control the offset. Setting the range of offset is equivalent to setting the range of denom.

Step 3: Perform transform and quantization on the residuals after changing denom. Consider 4×4 integral DCT and quantization. This process is the same as the transform and quantization process in H.264.

(a) Transform

$$Y = C_f X C_f^T \tag{23}$$

(b) Quantization

$$Z_{ij} = (Y_{ij} \cdot Q\,(QP\%6, i, j) + f) \gg (15 + QP/6) \tag{24}$$

(c) Inverse quantization

$$Y'_{ij} = \begin{cases} \left(Z_{ij} \cdot R\,(QP\%6, i, j)\right) \ll \left(\frac{QP}{6} - 4\right), QP \geq 24 \\ \left(Z_{ij} \cdot R\,(QP\%6, i, j) + 2^{3-\frac{QP}{6}}\right) \\ \qquad \gg \left(4 - \frac{QP}{6}\right), QP < 24 \end{cases} \tag{25}$$

(d) Inverse transform and normalization

$$X' = C_i Y' C_i^T \tag{26}$$

$$x_{ij} = (x'_{ij} + 2^5) \gg 6 \tag{27}$$

Step 4: Follow (28) to compute the PSNR of the frame given the current quantization offset. If the current PSNR is better than the previous best value, store the current offset factor denom and the value of PSNR.

$$PSNR = 10log_{10}\frac{(2^n - 1)}{MSE} \tag{28}$$

Step 5: Traverse all pre-set denom to obtain the optimal offset within this range.

The experimental results are available in Section V.

V. EXPERIMENTAL RESULTS AND DISCUSSION

In this section, this study compare the performance of the proposed algorithm, which adopts chrominance macroblock resizing and self-adaptive QP offset, with the original fractal algorithm. Then, this study compares the performance of the improved fractal algorithm with the state-of-the-art reference software JM19.0 of H.264/AVC. The algorithms are implemented on a computer with Intel Xeon E5-1620 v3 CPU and 32G memory. Some parameters of the program's configuration file are given in Table 2.

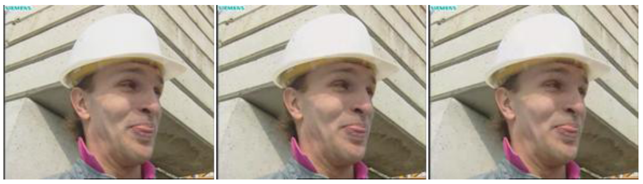

FIGURE 5. **Frames from different partition modes: (a) Original video; (b) 8×8-2×2 partitioning; (c) 16×16-4×4 partitioning.**

TABLE 3. **PSNR variation after changing partition mode.**

Speed	Test video	ΔPSNR(U) (dB)	ΔPSNR(V) (dB)
Slow	bridge-close	-0.21	-0.19
	foreman	-0.17	-0.14
	mobile	-0.15	-0.09
Moderate	coastguard	-0.18	-0.13
	flower	-0.1	-0.1
	paris	-0.14	-0.09
Fast	bus	-0.15	-0.12
	football	-0.14	-0.14
	stefan	-0.11	-0.14

TABLE 4. **Coding time variation after changing partition mode.**

Speed	Test video	T_{before} (ms)	T_{after} (ms)	ΔT (%)
Slow	bridge-close	7358	4441	-39.6439
	foreman	5578	5327	-4.49982
	mobile	11711	5099	-56.4597
Moderate	coastguard	9867	4662	-52.7516
	flower	7556	4394	-41.8475
	paris	6813	4609	-32.3499
Fast	bus	10936	6063	-44.5593
	football	9989	4277	-57.1829
	stefan	6204	5456	-12.0567

A. CHROMINANCE MACROBLOCK RESIZING

Firstly, this study compares the performance of the algorithm after the chrominance macroblock resizing. Generally, altering the chrominance block size causes slight decrease in the PSNR of U and V components, but the decrease is hardly discernable, as shown in Fig. 5. When QP=0, PSNR decreases by the largest margin of 0.1∼0.2db. In other cases, the decrease is less than 0.05db. Hence, it can be said that there is little degradation in the quality of image. Table 3 provides the variation of PSNR for U and V components in each test video, given QP=0.

The bitrate remains almost the same after the resizing. Only 11 of the 72 tests witness a variation of the bitrate by over 1%, and the variation is upper bounded by ±5% across all tests. However, the experimental results indicate that the coding time is shortened significantly. Table 4 shows the variation of the coding time for each video given QP=28.

TABLE 5. Coding time and PSNR variation after applying self-adaptive QP offset.

Speed	Test video	ΔT/%	ΔPSNR/dB
Slow	bridge-close	39.24792	1.02
	foreman	14.15431	0.85
	mobile	53.28496	0.94
Moderate	coastguard	38.20249	0.88
	flower	47.97451	0.97
	paris	38.64179	1.04
Fast	bus	12.0402	0.9
	football	38.5083	0.93
	stefan	12.02346	0.9

It can be observed that resizing the chrominance block substantially reduces the coding time without compromise of image quality and bitrate.

B. SELF-ADAPTIVE QP OFFSET

After resizing the chrominance block, this study introduces the self-adaptive QP offset algorithm. Experimental results indicate that the searching for the optimal QP offset results in an increase in the coding time, but the PSNR of the coded image is enhanced. In order to reduce the influence of optimal QP searching on time consuming, the proposed self-adaptive algorithm is only applied to the coding process of the luminance component. Another reason for this is that human is more sensitive to luminance change than to chrominance change [26]. Increasing PSNR of the luminance component promises more performance gains.

PSNR of the luminance component increases by the largest margin of about 4dB when QP=0. It can increase by about 1.4dB when QP=4 and by 0.9∼1dB when QP=8, 20, 28 and 36. Table 5 presents the variation of coding time and PSNR of the Y component for each video given QP=28.

C. PERFORMANCE COMPARISON WITH JM19.0

JM19.0 is the latest reference software of H.264/AVC. A comparison is made between the proposed algorithm and JM19.0 to demonstrate the performance of our algorithm. We evaluate the subjective quality first and then a comparison is made under the following metrics: PSNRs of the Y, U and V components, coding time and bitrate.

We choose 19th frame of the 'mobile', 'paris', and 'bus' when QP=14. Note that the 19th frame is the last P frame before the next I frame appears. The original frame, the reconstructed frame from JM19.0 and the reconstructed frame from the proposed algorithm are shown in Fig. 6.

It can be found that the reconstructed frames from the proposed algorithm have almost the same quality as that from the original video and JM19.0.

When encoding with a low QP, PSNR of the Y component in the proposed algorithm is smaller than that in the JM19.0 algorithm by 2dB at most. Due to the use of larger blocks, PSNRs of the U and V components decrease by a larger margin than that of the Y component, and are smaller than those of JM19.0 by about 8dB at most. But when encoding with a high QP, the value of PSNR in the proposed algorithm is higher than that in JM19.0. Human is more sensitive to luminance change than to chrominance change, so the PSNR decrease in chrominance components is acceptable. The bitrate – PSNR curve of the test video 'football' for Y, U and V component is shown in Fig. 7.

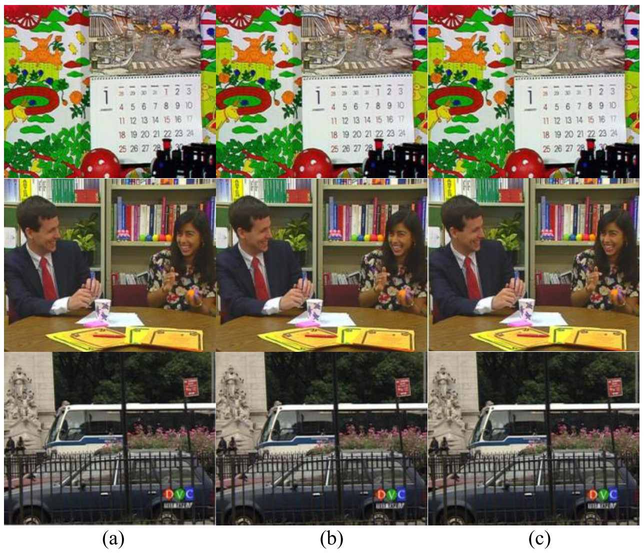

FIGURE 6. Reconstructed frames from: (a) Proposed algorithm; (b) JM19.0; (c) Original video.

TABLE 6. Coding time comparison of proposed algorithm and JM19.0.

Sequence	QP	T_{JM19} (ms)	$T_{Fractal}$ (ms)	ΔT (%)
bridge-close	0	13796	7089	-49.70
	14	12498	5530	-55.75
	28	10869	6184	-43.10
coastguard	0	14799	6057	-59.07
	14	13300	6855	-48.46
	28	12206	6443	-47.21
stefan	0	14871	5788	-61.08
	14	13339	6613	-50.42
	28	11952	6112	-48.86

From those figures, it can be observed that the proposed algorithm can make better use of the bitrate because its PSNR is higher than that of JM19.0 given a high bitrate. But the proposed algorithm is inferior to JM19.0 when the bitrate is low. A possible reason is that the proposed algorithm does not skip the macroblocks whose residuals are all zeros. Meanwhile, the fractal parameters transmitted occupy some bits and thus limit the scope for the bitrate to decrease.

The proposed algorithm achieves enormous superiority in terms of the coding time. Performance results of the algorithms are given in Table 6.

It can be seen that the proposed algorithm shortens the coding time by about 50% when compared with JM19.0 because of its low computation complexity. The coding time is much shorter than that of JM19.0 even after applying self-adaptive QP offset algorithm, which may increase the coding time by 10%-50%.

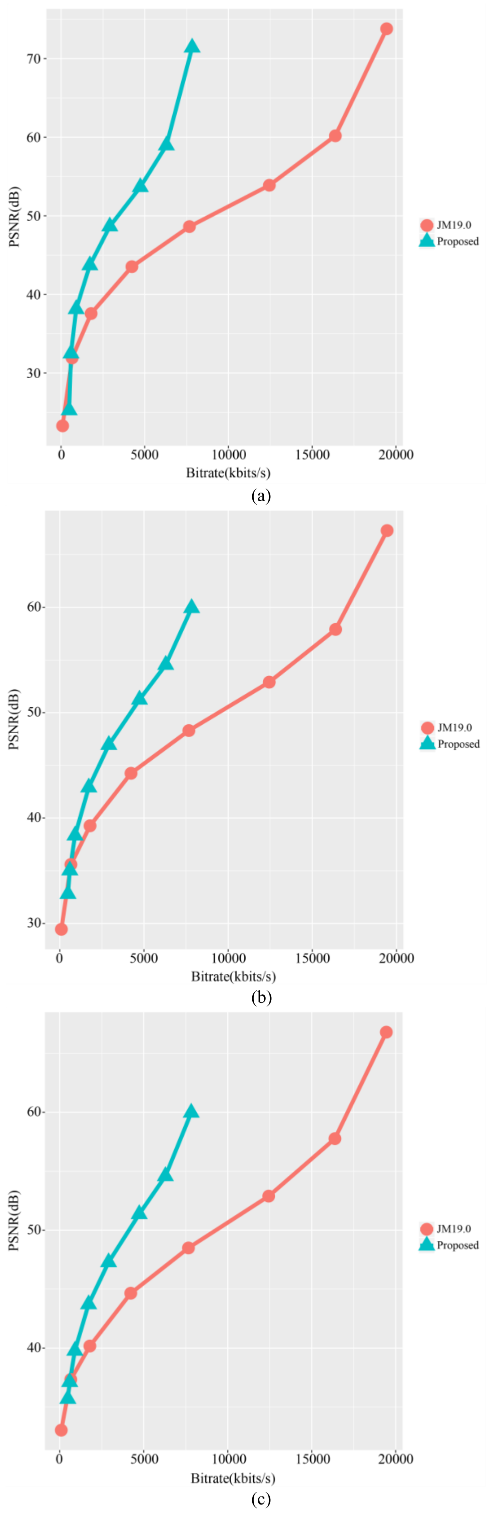

FIGURE 7. **Bitrate-PSNR curve: (a) football(Y); (b) football(U); (c) football(V).**

VI. CONCLUSION

In this paper, this study proposes an improved fractal video compression algorithm, which combines the intra-frame coding algorithm of H.264 and the fractal inter-frame coding algorithm. Given the same quantization parameters, the proposed algorithm can reduce the bitrate by more than 50% and shorten the coding time by over 45% in most cases, while guaranteeing that PSNR of the video is comparable to JM19.0. Performance comparison with JM19.0 indicates that the proposed algorithm is efficient and effective in moderate and high bitrate applications.

APPENDIX

The process to derive the fractal coefficients s and o:

The differences between the current frame and the reference frame is calculated through (29)

$$SSD = \sum_{i=1}^{N} |r_i - (s \cdot d_i + o)|^2 \tag{29}$$

where N denotes the number of pixels in the macroblock, r_i denotes the values of pixels in the range block, d_i denotes the values of pixels in the domain block. To minimize SSD, find the partial derivative of the square of SSD with respect to s and o:

$$\frac{\partial SSD(s,o)}{\partial s} = \sum_{i=1}^{N} 2(sd_i + o - r_i) \cdot d_i = 0 \tag{30}$$

$$\frac{\partial SSD(s,o)}{\partial o} = \sum_{i=1}^{N} 2(sd_i + o - r_i) = 0 \tag{31}$$

Then we have:

$$o = \frac{1}{\sum_{i=1}^{N} d_i}\left(\sum_{i=1}^{N} r_i d_i - s\sum_{i=1}^{N} d_i^2\right) \tag{32}$$

$$s\sum_{i=1}^{N} d_i = \sum_{i=1}^{N} r_i - N \cdot o \tag{33}$$

Furthermore, from Equation (32) and (33) we can finally find the equation to calculate s and o:

$$\begin{cases} s = \dfrac{N\sum\limits_{i=1}^{N} r_i d_i - \sum\limits_{i=1}^{N} r_i \sum\limits_{i=1}^{N} d_i}{N\sum\limits_{i=1}^{N} d_i^2 - \left(\sum\limits_{i=1}^{N} d_i\right)^2} \\ o = \dfrac{1}{N}\left[\sum\limits_{i=1}^{N} r_i - s\sum\limits_{i=1}^{N} d_i\right] \end{cases} \tag{34}$$

REFERENCES

[1] E. G. I. Richardson. (Mar. 2003). *H. 264/MPEG-4 Part 10 White Paper*. [Online]. Available: http://www.vcodex.com

[2] A. Puri, X. Chen, and A. Luthra, "Video coding using the H. 264/MPEG-4 AVC compression standard," *Signal Process., Image Commun.*, vol. 19, no. 9, pp. 793–849, 2004.

[3] G. Cote, B. Erol, M. Gallant, and F. Kossentini, "H.263+: Video coding at low bit rates," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, no. 7, pp. 849–866, Nov. 1998.

[4] S. Acharjee, D. Biswas, N. Dey, P. Maji, and S. S. Chaudhuri, "An efficient motion estimation algorithm using division mechanism of low and high motion zone," in *Proc. IEEE Int. Multi-Conf. Autom., Comput., Commun., Control Compressed Sens. (iMacs)*, Mar. 2013, pp. 169–172.

[5] M. Wang, R. Liu, and C.-H. Lai, "Adaptive partition and hybrid method in fractal video compression," *Comput. Math. Appl.*, vol. 51, no. 11, pp. 1715–1726, 2006.
[6] M. S. Lazar and L. T. Bruton, "Fractal block coding of digital video," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 4, no. 3, pp. 297–308, Jun. 1994.
[7] C.-M. Lai, K.-M. Lam, and W.-C. Siu, "A fast fractal image coding based on kick-out and zero contrast conditions," *IEEE Trans. Image Process.*, vol. 12, no. 11, pp. 1398–1403, Nov. 2003.
[8] M. F. Barnsley, "Fractals everywhere," *J. Roy. Statist. Soc.-Ser. A Statist. Soc.*, vol. 158, no. 2, p. 339, 1995.
[9] A. E. Jacquin, "Image coding based on a fractal theory of iterated contractive image transformations," *IEEE Trans. Image Process.*, vol. 1, no. 1, pp. 18–30, Jan. 1992.
[10] S. P. Zhu, Y. S. Hou, Z. K. Wang, and E. B. Kam, "Fractal video sequences coding with region-based functionality," *Appl. Math. Model.*, vol. 36, no. 11, pp. 5633–5641, 2012.
[11] S. G. Farkade and S. D. Kamble, "A hybrid block matching motion estimation approach for fractal video compression," in *Proc. Int. Conf. Circuits Power Comput. Technol. (ICCPCT)*, Mar. 2013, pp. 1151–1155.
[12] R. E. Chaudhari and S. B. Dhok, "Acceleration of fractal video compression using FFT," in *Proc. 15th Int. Conf. Adv. Comput. Technol. (ICACT)*, Sep. 2013, pp. 1–4.
[13] S. D. Kamble, N. V. Thakur, L. G. Malik, and P. R. Bajaj, "Fractal video coding using modified three-step search algorithm for block-matching motion estimation," in *Proc. Int. Conf. Comput. Vis. Robot., Adv. Intell. Syst. Comput. (ICCVR)*, vol. 332. 2015, pp. 151–162.
[14] S. D. Kamble, N. V. Thakur, L. G. Malik, and P. R. Bajaj, "Quadtree partitioning and extended weighted finite automata-based fractal colour video coding," *Int. J. Image Mining*, vol. 2, no. 1, pp. 31–56, 2016.
[15] S. P. Zhu, L. Li, J. Chen, and K. Belloulata, "An automatic region-based video sequence codec based on fractal compression," *AEU-Int. J. Electron. Commun.*, vol. 68, no. 8, pp. 795–805, 2014.
[16] S. D. Kamble, N. V. Thakur, and P. R. Bajaj, "A review on block matching motion estimation and automata theory based approaches for fractal coding," *Int. J. Interactive Multimedia Artif. Intell.*, vol. 4, no. 2, pp. 91–104, 2016.
[17] S. Acharjee, G. Pal, T. Redha, S. Chakraborty, S. S. Chaudhuri, and N. Dey, "Motion vector estimation using parallel processing," in *Proc. IEEE Int. Conf. Circuits, Commun., Control Comput. (IC)*, Nov. 2014, pp. 231–236.
[18] Z. Huang, "Frame-groups based fractal video compression and its parallel implementation in Hadoop cloud computing environment," *Multidimensional Syst. Signal Process.*, pp. 1–18, Mar. 2017, doi: 10.1007/s11045-017-0480-1.
[19] A. M. H. Y. Saad and M. Z. Abdullah, "High-speed implementation of fractal image compression in low cost FPGA," *Microprocess. Microsyst.*, vol. 47, pp. 429–440, Nov. 2016.
[20] A. M. H. Saad and M. Z. Abdullah, "Real-time implementation of fractal image compression in low cost FPGA," in *Proc. IEEE Int. Conf. Imag. Syst. Techn. (IST)*, Oct. 2016, pp. 13–18.
[21] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560–576, Jul. 2007.
[22] *H.264/AVC Reference Software JM 19.0*. Accessed: Jun. 30, 2017. [Online]. Available: http://iphome.hhi.de/suehring/tml/
[23] J. E. Hutchinson, "Fractals self-similarity," *Indiana Univ. Math.*, vol. 35, no. 5, pp. 713–741, 1981.
[24] Y. Fisher, *Fractal image Compression: Theory and Application*. New York, NY, USA: Springer, 2012.
[25] B. Sankaragomathi, L. Ganesan, and S. Arumugam, "Encoding video sequences in fractal-based compression," *Fractals*, vol. 15, no. 4, pp. 365–378, 2007.
[26] W. Wang, J. Dong, and T. Tan, "Effective image splicing detection based on image chroma," in *Proc. 16th IEEE Int. Conf. Image Process. (ICIP)*, Nov. 2009, pp. 1257–1260.

SHIPING ZHU (M'05) was born in Baoji, China, in 1970. He received the B.Sc. and M.Sc. degrees in measuring and testing technologies and instruments from the Xi'an University of Technology, Xi'an, China, in 1991 and 1994, respectively, and the Ph.D. degree in precision instrument and machinery from the Harbin Institute of Technology, Harbin, China, in 1997.

From 1997 to 1999, he was a Post-Doctoral Fellow with Beihang University, Beijing, China. From 2000 to 2002, he was a Post-Doctoral Fellow with the Brain and Cognition Research Center, Université Paul Sabatier, Toulouse, France. From 2002 to 2004, he was a Post-Doctoral Fellow with the Department of Computer Science and the Department of Electrical and Computer Engineering, Université de Sherbrooke, Sherbrooke, QC, Canada. Since 2005, he has been an Associate Professor with the Department of Measurement Control and Information Technology, School of Instrumentation Science and Optoelectronics Engineering, Beihang University. He has authored or coauthored over 80 journal and conference papers. He holds 50 China invention patents. His current research interests include image processing and video coding, computer vision, and machine vision for 3-D measurement.

SHUPEI ZHANG was born in Panzhihua, China, in 1994. He received the B.Sc. degree in instrumentation science and opto-electronics engineering from Beihang University, Beijing, China, in 2016, where he is currently pursuing the master's degree. His current research interests include video compression.

CHENHAO RAN was born in Yuncheng, China, in 1995. He received the B.Sc. degree in instrumentation science and opto-electronics engineering from Beihang University, Beijing, China, in 2016, where he is currently pursuing the master's degree. His current research interests include TDLAS and image reconstruction from histogram.