CHAPTER 3
ENCODING VIDEO SEQUENCES IN FRACTAL BASED COMPRESSION

3.1 Introduction

Day by day, the demands for higher and faster technologies are rapidly increasing. Although the technologies available now are considered to be more advanced than 30-40 years ago, people are still looking for improvements and enhancements. In the last twenty years, computers have been developed and their price has reached a level of acceptance where almost anyone can buy one. Nowadays, before purchasing computers, customers are concerned about two things: (1) the speed of the CPU; and (2) the storage and memory capacity. Image and video compression helps to reduce the memory capacity and to have faster transmission rates. Many compression techniques [117, 118] have been introduced and developed, such as Joint Photographic Experts Group (JPEG) and Graphics Interchange Format (GIF) for image compression as well as Moving Picture Experts Group (MPEG) for video compression [102]. Although some of these techniques show significant and enhanced performance in decreasing the cost of the transmission and storing data, the search for alternatives continues. Fractal image or video compression [85] is a new compression method which is based on self-similarity within the different portions of the image. It might be revolutionary in the world of data compression because of its high compression rate compared with other methods; however, it suffers from some problems such as the time taken for encoding [93].

Fractal dimension is a promising feature proposed to characterize roughness and self similarity in an image sequence. Roughness usually results from the edge components in the spatial domain and movement in the temporal domain, while self similarity corresponds to both spatial and temporal redundancy. The above observation
serves as a motivation for employing fractal dimension as the major feature discriminator. The theory of iterated contractive transformations is utilized to record the transformation function for every partition between the domain and range block [94, 95, 96]. However, the results reported, 0.68 bits/pixel compression rate (11.76 compression ratio) with SNR 27.7 dB in the two dimensional case and 41.80 and 74.39 compression ratio with PSNR around 29dB and 33dB, respectively in the 3-dimensional case. In order to meet the compression requirements of the diverse image sequence characteristics, a robust technique proposed is based on fractal dimensions of the ensemble luminance and chrominance, respectively, relative to a reference frame inside a Group Of Pictures (GOP). The method proposed is tested on image sequences containing various motion dynamics. The performance in terms of the compression ratio is tabulated and the PSNR value in all three color channels for each reconstructed frame is illustrated.

### 3.2 Fractal Color Image Compression

Little work has been done of fractal color image compression compared with the work done on fractal gray-scaled image compression [48]. The need for color image compression is gaining importance in recent times due to large scale multimedia applications. Conventional fractal compression schemes can easily be extended to color image compression as a color image is usually represented in multi channels such as Red, Green and Blue (RGB) components [118]. Thus each channel in color image can be compressed as a gray-level image. Hurtgen, Mols and Simon [63] proposed a fractal transform coding of color images. This kind of encoding lacks the possibility of considering similarities between the three color planes, thus a good compression ratio is not achieved. Fair population on the domain pool is needed to obtain a high quality interpolation for zooming purposes. To exploit the spectral
redundancy in RGB components, the root mean square error (RMS) measure in grayscale space can be extended to 3-dimensional color space for fractal-based color image coding [29]. Experiments show that a 1.5 compression ratio improvement can be obtained using vector distortion measure in fractal coding with fixed image partition as compared to separate fractal coding in RGB images. However, since RGB space is not perceptually uniform, it is decided to use another color space, called CIE-Lab.

**The color space**

A color space is a mathematical representation of a set of colors. The three most popular color models are RGB (used in computer graphics), YIQ, YUV or YCbCr (used in video systems) and CMYK (used in color printing). However, none of these color spaces are directly related to the intuitive notions of hue, saturation and brightness, which are the basis of our color perception. This resulted in the temporary pursuit of other models, such as HSI and HSV, to simplify programming, processing and end-user manipulation but trying to get closer to the actual representation of colors in our brain. Indeed, mathematically, all the color spaces can be derived from the RGB information supplied by devices such as cameras and scanners. The (RGB) color space is widely used throughout computer graphics, since red, green and blue are three primary additive colors (individual components are added together to form a desired color) and are represented by a three-dimensional, Cartesian coordinate system [80].

The triangle, called Maxwell triangle has been drawn between the three primaries. The intersection point of a color vector with the triangle gives an indication of the hue and saturation of the color in terms of the distances of the point from the vertices of the triangle.
The RGB color space is the most prevalent choice for computer graphics because color displays use red, green and blue to create the desired color. Therefore, the choice of the RGB color space simplifies the architecture and design of the system. Also, a system that is designed using the RGB color space can take advantage of a large number of existing software routines, since this color space has been around for a number of years. However, RGB is not very efficient when dealing with “real-world” images. All three RGB components need to be of equal band width to generate any color within the RGB color cube.

The result of this is a frame buffer that has the same pixel depth and display resolution for each RGB component. Also, processing an image in the RGB color space is usually not the most efficient method. For example, to modify the intensity or color of a given pixel, the three RGB values must be read from the frame buffer, the intensity or color calculated, the desired modifications performed and the new RGB values calculated and written back to the frame buffer. If the system has access to an image stored directly in the intensity and color format, the processing steps would be faster. The most common systems are the YUV and YCbCr color spaces. The YUV color space is used by the PAL (Phase Alternation Line), NTSC (National Television...
The equations that describe the direct transformation RGB to YUV are:

\[
Y = 0.299 \, R + 0.587 \, G + 0.114 \, B
\]

\[
U = -0.147 \, R + 0.289 \, G + 0.436 \, B = 0.492 \, (B - Y)
\] (3.1)

\[
V = 0.615 \, R - 0.515 \, G - 0.100 \, B = 0.877 \, (R - Y)
\]

and for the inverse transformation: \( R \, Y \, V \)

\[
R = Y + 1.140 \, V
\]

\[
G = Y - 0.395 \, U - 0.581 \, V
\] (3.2)

\[
B = Y + 2.032 \, V
\]

For digital RGB values with a range of 0 - 255, Y has a range of 0 - 255, U a range of 0 to ±112 and V a range of 0 to ±157. As the RGB color space, the YUV space is not uniform concerning the HVS. A system is said to be not uniform if a little perturbation of a value is perceived linearly along the possible variation of that value. Using a non-perceptually uniform space as RGB has the drawback that the Human Vision System will be affected by computer measures for digital video processing, since the distance from RGB value will not be uniform in respect of the HVS. Starting from these considerations, the Commission Internationale d’Eclairage (CIE) defined a uniform color model. Danciu and Hart [29] presented a comparative study of fractal color image compression in the CIE-Lab color space with that of Jacquin's iterated transform technique for 3-dimensional color. It has been shown that the use of uniform color space yields compressed images to have less noticeable color distortion than other methods. Since there are three types of color photoreceptor cone cells in the retina, each with a different spectral response curve, all colors can be completely described by three numbers, corresponding to the outputs of the cone cells. The CIE workgroup then defined XYZ tristimulus values, where all visible colors can be represented using only positive values of
X, Y and Z. For applications where it is important to be able to measure differences between colors in a way that matches perceptual similarity as good as possible, the perceptually uniform color spaces find their best field of use. The CIE-Lab color space is designed such that the perceived differences between single, nearby colors correspond to the Euclidean distance of the color coordinates. The (nonlinear) conversions from RGB to CIE-Lab are given by:

$$
\begin{bmatrix}
X \\
Y \\
Z
\end{bmatrix} =
\begin{bmatrix}
0.412453 & 0.357580 & 0.180423 \\
0.212671 & 0.715160 & 0.072169 \\
0.180423 & 0.357580 & 0.412453
\end{bmatrix}
\begin{bmatrix}
R \\
G \\
B
\end{bmatrix}
$$

(3.3)

Figure 3.2 CIE-Lab color space representation

Luminance varies from 0 (black) to 100 (white) and ’a’ and,’b’ components vary from −50 to +50 and represent the color variation along the red-green and blue-yellow axis.

The two-dimensional fractal encoder works on gray-level images, or with one layer of color decomposition. For encoding a color image, it should be decomposed in three color layer decomposition as separated RGB or YIQ layers [72].
Every image will be partitioned in squared domain and range blocks. It is assumed that:

- **Range blocks have a constant size equal to** $R$
- **Domain blocks have a constant size equal to** $D$, which is two times the size of a range block.
- **The Domain Pool is a “Non Overlapped Domain Block”**.

Given a starting image $\mu_{\text{orig}}$ the encoder process will be for Range Block $R_i$ on the partition obtained from the starting image $\mu_{\text{orig}}$:

1) extract the Range Block $R_i$;
2) finding the fractal transform of this block;
3) saving the fractal code $\tau_i$

The procedure involved in the step (2) is given below
2.a) from the input Range Block $R_i$ computing its luminance $l_{R_i}$ and its contrast $C_{R_i}$.

The luminance $l_{R_i}$ and the contrast $C_{R_i}$ of the range block $R_i$ is computed the following way:

- **Luminance $l_{R_i}$** is the minimum value of luminance among all pixels that compose $R_i$:

  $$l_{R_i} = \min_{(x,y) \in R_i} [p_{R_i}(x,y)]$$

  where $p_{R_i}(x,y)$ is a generic pixel of $R_i$ whereas $R$ is the size of the Range Block.

- **Contrast $C_{R_i}$** is obtained by the overall sum of pixels inside the Range $R_i$. Before the addition, the previous luminance $l_{R_i}$ is subtracted to every pixel’s value.

2.b) computing $R_i$ with the norm of the Range Block using the luminance and the contrast. For every Domain Block $D_j$ inside the Domain Pool:

The normalization $|R_i|$ of the range block $R_i$ is obtained:

- From the pixels that belong to the Range Block $R_i$, the luminance $l_{R_i}$ is subtracted (luminance shift).

- Every pixel is then divided by the contrast $C_{R_i}$ (contrast scaling).

- It is easy to notice that $l_{R_i} = 4$. To obtain the contrast we subtract the luminance value to every pixel, and then we sum all the pixels together:

  $$C_{R_i} = (10 - 4) + (7 - 4) + (5 - 4) + (4 - 4) = 10 \quad (3.4)$$
2.c) sub sampling $D_j$ to match Range Block’s size.

- The generic domain block ‘$D_j$’ is subsampled by a factor of two stated as a condition) to have the same dimension of the range block ‘$R_i$’
- The same procedure of 2.a) is now executed on ‘$D_j$’
- The same procedure of 2.b) is now executed on ‘$D_j$’

2.d) computing its luminance $l_{D_i}$ and its contrast $C_{D_i}$.

2.e) computing $|D_j|$ with the norm of $D_j$ using $l_{D_i}$ and $C_{D_i}$.

An isometry ‘$k$’ is chosen.

2.f) computing the isometry $|D_j|_k$ starting from $|D_j|$.

2.g) computing the error between $|R_i|$ and $|D_j|_k$.

A metric is needed to compare ranges and transformed domain. The simple MSE between pixels of ‘$|R_i|$ and $|D_j|_k$’ is used as a first approximation

$$MSE( |R_i|, |D_j|_k) = \frac{1}{R \cdot R} \sum_{x=1}^{R} \sum_{y=1}^{R} \left[ P_{|R_i|}(x,y) - P_{|D_j|_k}(x,y) \right]^2 \tag{3.5}$$

2.h) if the error is a local minimum, save $\tau_i$ composed by information on isometry and domain used. Finally the fractal code $\tau_i$ of the Range Block $R_i$ is obtained considering:

- The coordinates $\left(X_{D_j}, Y_{D_j}\right)$ that identify the Domain Block $D_i$ (founded as in 2.g that minimize the MSE error measure) on the encoding image.
- The isometry $k$ that minimize the error measure.
Luminance $l_{R_i}, l_{D_j}$ and the contrast ratio:

2.i) end “isometry” loop

end “Domain Block” loop

![Figure 3.4 An example of normalization: a) the range block b) a three-dimensional representation of the range block c) luminance shift d) contrast.](image)

Allowed isometries are

<table>
<thead>
<tr>
<th>Identity</th>
<th>$\tau_1(\mu_{i,j}) = \mu_{i,j}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Horizontal reflection</td>
<td>$\tau_2(\mu_{i,j}) = \mu_{i,n-j}$</td>
</tr>
<tr>
<td>Vertical reflection</td>
<td>$\tau_3(\mu_{i,j}) = \mu_{n-i,j}$</td>
</tr>
<tr>
<td>First diagonal reflection</td>
<td>$\tau_4(\mu_{i,j}) = \mu_{j,j}$</td>
</tr>
<tr>
<td>Second diagonal reflection</td>
<td>$\tau_5(\mu_{i,j}) = \mu_{n-j,n-i}$</td>
</tr>
<tr>
<td>90° counter-clockwise rotation</td>
<td>$\tau_6(\mu_{i,j}) = \mu_{n-j,i}$</td>
</tr>
<tr>
<td>180° counter-clockwise rotation</td>
<td>$\tau_7(\mu_{i,j}) = \mu_{n-i,n-j}$</td>
</tr>
<tr>
<td>270° counter-clockwise rotation</td>
<td>$\tau_8(\mu_{i,j}) = \mu_{j,n-i}$</td>
</tr>
</tbody>
</table>

$$C_j = \frac{C_{R_i}}{C_{D_j}}$$

All these steps are mandatory, to obtain an efficient matching between Range Blocks $R_i$ and all the isometries of the subsampled Domain Block $D_j$. The overall process is explained using the given range block as an example.

<table>
<thead>
<tr>
<th>10</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>
Given the following Range Block and Domain Block (the Domain Block is already sub-sampled to match the Range Block size), it is noticed that they seem completely different from each other for every isometry applied to D. The luminances of the blocks are: \( l_{R_i} \) and \( l_{D_j} \). As stated in 2.b.1 subtracting the luminance to their respective blocks:

\[
\begin{array}{|cc|}
\hline
R_i & D_j \\
\hline
10 & 19 \\
7 & 13 \\
5 & 9 \\
4 & 7 \\
\hline
\end{array}
\]

\[
\begin{array}{|cc|}
\hline
R_i - l_{R_i} & D_j - l_{D_j} \\
\hline
10 & 10 \\
7 & 7 \\
5 & 5 \\
4 & 4 \\
\hline
\end{array}
\]

The contrast of the blocks are computed and applying the ratio as in 2.h:

\[
\frac{(R_i - l_{R_i})}{C_{R_i}} = \frac{(D_j - l_{D_j})}{C_{D_j}}
\]

\[
\begin{array}{|cc|}
\hline
0.6 & 0.6 \\
0.3 & 0.3 \\
0.1 & 0.1 \\
0 & 0 \\
\hline
\end{array}
\]

\[
l_{R_i} = 4 \\
l_{D_j} = 4
\]

\[
C_{R_i} = 10 \\
C_{D_j} = 20
\]

It is seen that the initial blocks that seemed different are now identical. All the operations made are reversible.
Bidimensional Fractal Decoder

The fractal code $\tau$ is computed with the method described above, a fractal approximation of its initial attractor (the initial image) $\mu_{\text{orig}}$ is re-obtained. It is assumed that:

- Range Blocks have a constant size of $R' = \beta \cdot R$
- Domain Blocks have a constant size $D'$ which is two times the size of the Range Blocks
- The starting image of the decoding process $\mu_0$ has a size that is $\mu_0 \cdot \beta$ times the size of the initial encoded image $\mu_{\text{orig}}$

The factor $\beta$ is the zooming factor used at the decoding stage to obtain variable different sizes of decode image. Given the fractal code $\tau$ and a starting image $\mu_0$, the decoding stage consists of the following steps:

1) for $m < \alpha$ iterations

2) for every $\tau_i$ of $\tau$

3) reading from $\tau_i$ the domain block coordinates $\left(x_{D_i}, y_{D_i}\right)$

4) extracting the Domain Block $D_i$ situated at the position $\left(\beta x_{D_i}, \beta y_{D_i}\right)$ of $\mu_m$

5) Computing a sub sample approximation $D_{i}^{\text{sub}}$ of $D_i$ to match the size of $R_i$

6) applying the $k$ isometry extracted from $\tau_i$ to $D_{i}^{\text{sub}}$ obtaining $D_{i}^{\text{sub}}_{k}$

7) extracting $l_{R_i}, l_{D_i}$ and $c_{i}$ from $\tau_i$.

8) applying luminance values and contrast to $D_{i}^{\text{sub}}_{k}$ obtaining $R_{i}^{\text{decoded}}$

9) writing $R_{i}^{\text{decoded}}$ on $\mu_{m+1}$
End $\tau$, loop

End iteration loop

The previous steps mean:

1) This loop cycles the whole decoding procedure $\alpha$ times: this is the
   cycle that generates the sequence of images that will converge
   towards the attractor ($\mu_{\text{orig}}$ or an expanded version of it).

2) This loop decodes all the $\tau$ fractal transform related to the Range
   Block $R_i$.

3) Reads from $\tau$ the coordinates $(x_{D_i}, y_D)$ of Domain Block $D_i$.
   This
   domain is the best transformed domain that the encoder found in the
   original image $\mu_{\text{orig}}$ for $R_i$.

4) Since all the image is zoomed by a factor $\beta$, all the coordinates must be
   shifted of the same amount.

3.3 Video Compression

Video is a number of images in sequence; therefore, the same method of
compressing images can be applied on the video by compressing each frame of the
video separately (this is called intra-frame coding). Though this technique looks very
simple, it is impractical because it requires a very large memory space to store data.
The best way to achieve a better compression is in taking advantage of the similarities
between the video frames. There are two main functions used in video coding [80].

1. Prediction: create a prediction of the current frame based on one or
   more previously transmitted frames.

2. Compensation: subtract the prediction from the current frame to
produce a ‘residual frame’.

### 3.3.1 Frame Differencing

The idea of frame differencing in video coding is to produce a residual frame by subtracting the previous frame from the current frame; the residual frame will then be a form of zero data, with light and dark areas (light indicates positive residual data and dark indicates negative residual data). Nevertheless, there will be more portions in the residual frame with zero data than the light and dark; this is due to the similarity between the frames (most of the pixels in the previous frame will be equal to the pixels in the current frame). Therefore, since more of the residual frame is zero data, then the compression efficiency will be further improved, if the residual frame is compressed instead of the current frame. Frame differencing method in video coding faces a major problem which is best illustrated by the following example. In the encoding and the decoding process there will be no prediction for the first frame, but the problem starts with the second frame when the encoder uses the first frame as the prediction and encodes the resulting residual frame. The decoded first frame is not exactly the same as the input frame, which leads to a small error in the prediction of the second frame at the decoder. This error will increase as it is continued with other frames and the result will be of low quality in the decoded video sequence [92, 99].

### 3.3.2 Motion-compensation Prediction

When the differences between the previous frame and the current frame are not really similar or if there is a big change between the frames, then the compression may not be significant. This is due to the movement in the video scene. So, in this type of frames, another method of prediction used is called Motion-compensation Prediction where the achievement of better prediction is by estimating the movement and
compensating for it. Motion-compensation is similar to frame differencing with two extra steps.

1. **Motion estimation**: Comparing a region in the current frame with the neighboring regions of the previous decoded frame and finding the best match.

2. **Motion Compensation**: Subtracting the matching region from the current region. The encoder will send the location of the best match to the decoder to perform the same motion compensation operation in the process of decoding the current frame.

   The residual frame in motion-compensation contains less data compared with frame differencing (higher compression); however, motion-compensation is computationally very intensive.

### 3.4 Fractal Video Compression

For fractal video compression, there are two extensions of still image compression [83]. They are frame-based compression and cube-based compression. In frame-based compression, video clips and motion pictures are naturally divided into segments according to scene changes. Each segment, beginning with an initial frame, is called an intra-coded frame, or I-frame. Each frame then can be coded mainly using the motion codes by referencing its preceding frame called a P-frame, as a predicted frame from its predecessor. The I-frames and the P-frames are also called coarse frames and the frames that are added between any two of the I-frames and P frames are called bidirectional frames or B-frames (Figure 3.5). Each B-frame is coded using the prediction from both coarse frames immediately before and after it.

In a 2-dimensional fractal video compression system, the I-frames are compressed using image compression technique. The I-frames are coded by 2-to-1
local and global self referencing fractal codes. While decoding such an I-frame, a hidden frame has been created in each iteration. For example, if an I-frame ‘F’ is created by ten iterations from some initial image ‘F₀’ with self reference fractal codes, ten consecutive P-frames will be set that have the same set of codes and are identical to the I-frame fractal codes, but will be referenced to the preceding frame instead of itself. Then, starting from the same initial image F₀, by the end, the tenth P-frame is clearly the same I-frame obtained in the first procedure. As a result, a fractal represented I-frame can be replaced by a sequence of P-frames if a time delay is allowed.

![Figure 3.5 Video clip frames](image)

In a cube based compression, image sequences are partitioned into groups of frames, and every group of frames is partitioned into non-overlapped cubes of ranges and domains. The compression codes are computed and stored for every cube. Every group of frames is called GOF. Each GOF can be compressed and decompressed separately as an entity. Assuming temporal axis along the sequence, every GOF can be considered as a large cuboid. In fractal compression, each GOF is partitioned into nonoverlap small cuboids. Each cuboid is called as a range cuboid and denoted as ‘R’. The sizes of edges of ‘R’ may be different especially the edge in the temporal direction may vary from the horizontal direction and the vertical direction. In order to obtain the
approximate transformation of ‘R’, another overlap partition is necessary whose small parts are called domain cuboids. The horizontal and the vertical edges of the domain cuboids are twice as large as the range cuboids respectively. But the temporal edge of the domain cuboid is the same as the one of the range cuboids. The cuboid algorithm of fractal video compression is given below:

i. Partitioning the motion image sequence to a series of GOF. For each GOF the following steps have been done:

ii. Partitioning the GOF into range cuboids and domain cuboids. The horizontal and the vertical edges of the domain cuboids are twice as large as the ones of the range cuboids respectively. The temporal edge of the domain cuboids is the same size as the one of the range cuboids.

iii. For each range cuboid ‘R’ the following steps have been done:

a) All domain cuboids are the same sizes as ‘R’ in three directions.

b) Computing the scale factor and the offset factor \( \alpha, \beta \) of ‘D’ and the rms error between ‘R’ and \( \alpha D + \beta I \).

c) Choosing the optimal approximation \( R \approx \alpha D + \beta I \) that have the minimal rms error.

d) Storing \( \alpha, \beta \) and the location of ‘D’ of the optimal approximation as the Compression codes of ‘R’.

The frame-based compression can obtain high compression ratio, but the compression of the current frame is related to the previous decompressed image, so there is a delay between frames when decompressed and an error may spread between frames. The cube-based method can obtain the decompressed images with high qualities. However, if transmission error is considered, the adaptive partition is not suitable because the partition information may be lost during transmission.
If a video sequence is composed by ‘n’ frames:

\[ S = \bigcup_{i=1}^{n} \mu_i \] (3.9)

The encoded stream by just searching for appropriate transformations \( \mu_i \) on every single frame is obtained, so that the output video is:

\[ S' = \bigcup_{i=1}^{n} \tau(\mu_i) \] (3.10)

\[ \lim_{n \to \infty} S' = S \]

Being \( \square \) an initial arbitrary frame. In fact most of the frames are correlated to each other along time. To exploit this redundancy, another technique can be used to encode fractally a video. For every frame ‘i + 1’ the domain blocks are searched using the frame ‘i’ as the searching pool. The video sequence is first analyzed to find which frames are highly correlated to each other. An MSE ratio is used to divide the sequence into group of pictures (GOP) and then use the first frame as a domain pool for the rest of the GOP frames. Being the GOP composed by ‘p’ frames, a subsequence is encoded stating:

\[ S_m' = \bigcup_{i=1}^{p} \tau_i(\mu_0) \] (3.11)

\[ \tau_i : D_i \to R_i \]

The complete sequence will be

\[ S' = \bigcup_{m=1}^{j} S_m' \] (3.12)

Every frame in the packet is partitioned in range blocks, and a transformation of a domain block obtained by the first frame of the packet is chosen to be the best approximation of every range block on the current frame. At decoding time, the process is inverted and starting from a blank frame, all the other frames of a packet
are reconstructed using the transformation set obtained during the encoding stage. The
direct extension of the two-dimensional fractal encoding is to consider the sequence of
frames, i.e. the overall sequence, as a three-dimensional object with the third axis
represented by the timing. In fractal video coding, using the three-dimensional
extension range and domain blocks become three dimensional objects: range and
domain cubes. The process is straight-forward: the video sequence is partitioned into
range and domain cubes, and for every range cube, a transformed domain cube is
searched to minimize the error measure and to be the best approximation of it.

3.5 Experiments and Results

The proposed method has been applied to four video sequences such as
"Foreman" (144x144, 300 frames, 30 frames per second (fps)), "Mobile" (144x144, 300
frames, 30 fps), "Mother_daughter" (144x144, 300 frames, 30 fps) and "Suzie"
(144x144, 150 frames, 30 fps). The test sequences are given as input to the video
encoder and the parameters such as CR and PSNR value are calculated for each
sequence. The CR can be calculated as shown in below.

\[
CR = \frac{\text{original\_file\_size} - \text{compressed\_file\_size}}{\text{original\_file\_size}}
\]  
(3.13)

The video coding system uses PSNR value to assess the video quality, which is
illustrated in the equation given below.

\[
PSNR = 10\log_{10}\left(\frac{255^2}{\text{MSE}}\right)\ dB
\]  
(3.14)

DCT-based transformation (used in the existing transformation) and DWT-
based transformations such as Haar filter and Fractal transforms for the YUV sequences
such as Foreman, Mobile, Mother_daughter, and Suzie have been used. The
performance parameters measured include the PSNR values in each case along with
compression ratio, bits per frame and encoding time values. The PSNR value, compression ratio, encoding time, and average bits per frame of DCT, DWT and Fractal for "Foreman" and "Mobile" video sequences are compared in Table 3.1. The results show that our proposed technique is 1.13 to 1.195 times faster than the existing algorithm. As per the Luminance PSNR value is concerned, the proposed technique achieves a 2.07 to 2.5 dB improvement. Comparable performance is also obtained with the existing technique on CR.

**Table 3.1 Performance comparisons of DCT–based, DWT-based and Fractal-based transformations for "Foreman" and "Mobile" video sequences**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Video sequence</th>
<th>Foreman</th>
<th>Mobile</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>DCT</td>
<td>DWT</td>
</tr>
<tr>
<td>PSNR Y (dB)</td>
<td></td>
<td>33.68</td>
<td>35.75</td>
</tr>
<tr>
<td>PSNR U (dB)</td>
<td></td>
<td>33.80</td>
<td>36.19</td>
</tr>
<tr>
<td>PSNR V (dB)</td>
<td></td>
<td>33.87</td>
<td>36.00</td>
</tr>
<tr>
<td>Encoding time (sec)</td>
<td></td>
<td>1676.104</td>
<td>1451.217</td>
</tr>
<tr>
<td>Compression Ratio</td>
<td></td>
<td>0.8768</td>
<td>0.8768</td>
</tr>
<tr>
<td>Average bits/frame</td>
<td></td>
<td>11245624</td>
<td>11176528</td>
</tr>
</tbody>
</table>

**Table 3.2 Performance comparisons of DCT-based, DWT-based and Fractal-based transformations for "Mother_daughter" and "Suzie" video sequences**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Video sequence</th>
<th>Mother_daughter</th>
<th>Suzie</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>DCT</td>
<td>DWT</td>
</tr>
<tr>
<td>PSNR Y (dB)</td>
<td></td>
<td>34.48</td>
<td>36.76</td>
</tr>
<tr>
<td>PSNR U (dB)</td>
<td></td>
<td>34.13</td>
<td>36.19</td>
</tr>
<tr>
<td>PSNR V (dB)</td>
<td></td>
<td>34.18</td>
<td>36.23</td>
</tr>
<tr>
<td>Encoding time (sec)</td>
<td></td>
<td>1554.032</td>
<td>1183.067</td>
</tr>
<tr>
<td>Compression Ratio</td>
<td></td>
<td>0.9135</td>
<td>0.9135</td>
</tr>
<tr>
<td>Average bits/frame</td>
<td></td>
<td>7851616</td>
<td>7766600</td>
</tr>
</tbody>
</table>

Comparison of PSNR value, CR, encoding time, and average bits per frame.
of DCT, DWT and Fractal for "Mother_daughter" and "Suzie" video sequences is shown in Table 3.1. The results show that this technique is 1.098 to 1.314 times faster than the existing algorithm. Comparable performance in Compression Ratio is also obtained with the existing technique. Moderate improvements are also derived for Chrominance PSNR value. The proposed technique achieves a 1.098 dB to 1.314 dB improvement in Luminance PSNR value and this is shown in Table 3.2. Considerable improvements are also obtained for Chrominance PSNR value. Performance comparison of average bits required for motion information is given in Table 3.3. Performance comparison of average MSE per pixel required for motion information is represented in Table 3.4. Average number of search points per block for Car-phone video sequence is shown in Figure 3.6. The Performance comparisons of DCT-based, DWT-based and Fractal-based transformation for “Mobile” and Foreman video sequences are shown in Tables 3.5 and 3.6.

Table 3.3 Performance comparison of average bits required for motion information

<table>
<thead>
<tr>
<th>Video Sequence</th>
<th>Full search</th>
<th>Fast search</th>
<th>Proposed algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Foreman</td>
<td>1783.59</td>
<td>1654.1</td>
<td>1642.86</td>
</tr>
<tr>
<td>Container</td>
<td>371.67</td>
<td>357.69</td>
<td>352.96</td>
</tr>
<tr>
<td>Claire</td>
<td>478.29</td>
<td>459.14</td>
<td>434.46</td>
</tr>
<tr>
<td>Car-phone</td>
<td>1749.2</td>
<td>1613.77</td>
<td>1591.02</td>
</tr>
<tr>
<td>Miss America</td>
<td>469.97</td>
<td>459.48</td>
<td>450.76</td>
</tr>
<tr>
<td>Mobile</td>
<td>2725.27</td>
<td>2645.25</td>
<td>1642.86</td>
</tr>
<tr>
<td>Mother-daughter</td>
<td>755.67</td>
<td>722.13</td>
<td>721.01</td>
</tr>
<tr>
<td>News</td>
<td>917.43</td>
<td>848.69</td>
<td>827.76</td>
</tr>
<tr>
<td>Salesman</td>
<td>671.48</td>
<td>633.08</td>
<td>634.46</td>
</tr>
<tr>
<td>Silent</td>
<td>1009.31</td>
<td>946.76</td>
<td>931.47</td>
</tr>
<tr>
<td>Suzie</td>
<td>1115.64</td>
<td>1059.01</td>
<td>1039.62</td>
</tr>
</tbody>
</table>

The performance evaluations show that the improved transformation technique is 1.13 to 1.195 times faster than the existing algorithm. As per the Luminance PSNR value is concerned, the proposed technique achieves a 2.07 to 2.51
dB improvement. Improved CR is also obtained compared with the existing technique.

### Table 3.4 Performance comparison of average MSE per pixel required for motion information

<table>
<thead>
<tr>
<th>Video Sequence</th>
<th>Full search</th>
<th>Fast search</th>
<th>Proposed algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Foreman</td>
<td>2.5601</td>
<td>2.5621</td>
<td>2.5622</td>
</tr>
<tr>
<td>Container</td>
<td>2.5602</td>
<td>2.5641</td>
<td>2.5631</td>
</tr>
<tr>
<td>Claire</td>
<td>2.5610</td>
<td>2.5651</td>
<td>2.5633</td>
</tr>
<tr>
<td>Car-phone</td>
<td>2.5612</td>
<td>2.5633</td>
<td>2.5622</td>
</tr>
<tr>
<td>Miss America</td>
<td>2.5602</td>
<td>2.5606</td>
<td>2.5612</td>
</tr>
<tr>
<td>Mobile</td>
<td>2.5621</td>
<td>2.5611</td>
<td>2.5613</td>
</tr>
<tr>
<td>Mother-daughter</td>
<td>2.5611</td>
<td>2.5612</td>
<td>2.5611</td>
</tr>
<tr>
<td>News</td>
<td>2.5605</td>
<td>2.5651</td>
<td>2.5623</td>
</tr>
<tr>
<td>Salesman</td>
<td>2.5604</td>
<td>2.5621</td>
<td>2.5618</td>
</tr>
<tr>
<td>Silent</td>
<td>2.5610</td>
<td>2.5677</td>
<td>2.5624</td>
</tr>
<tr>
<td>Suzie</td>
<td>2.5602</td>
<td>2.5612</td>
<td>2.5633</td>
</tr>
</tbody>
</table>

### Figure 3.6 Average number of search points per block for

(a) Car-phone video sequence    (b) Container video sequence

### Table 3.5 Performance comparisons of DCT-based, DWT-based and Fractal-based transformations for “Foreman” video sequence

<table>
<thead>
<tr>
<th>Parameters</th>
<th>DCT</th>
<th>DWT</th>
<th>Fractal</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSNR Y (dB)</td>
<td>33.68</td>
<td>35.75</td>
<td>35.78</td>
</tr>
<tr>
<td>PSNR U (dB)</td>
<td>33.80</td>
<td>36.19</td>
<td>36.03</td>
</tr>
<tr>
<td>PSNR V (dB)</td>
<td>33.87</td>
<td>36.00</td>
<td>36.07</td>
</tr>
<tr>
<td>Encoding time (sec)</td>
<td>1676.104</td>
<td>1451.217</td>
<td>1380.325</td>
</tr>
<tr>
<td>Compression Ratio</td>
<td>0.8768</td>
<td>0.8768</td>
<td>0.8768</td>
</tr>
<tr>
<td>Average bits/frame</td>
<td>11245624</td>
<td>11176528</td>
<td>11166248</td>
</tr>
</tbody>
</table>
### Table 3.6 Performance comparisons of DCT-based, DWT-based and Fractal-based transformations for “Mobile” video sequence

<table>
<thead>
<tr>
<th>Parameters</th>
<th>DCT</th>
<th>DWT</th>
<th>Fractal</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSNR Y (dB)</td>
<td>32.05</td>
<td>34.48</td>
<td>34.56</td>
</tr>
<tr>
<td>PSNR U (dB)</td>
<td>31.89</td>
<td>34.47</td>
<td>34.55</td>
</tr>
<tr>
<td>PSNR V (dB)</td>
<td>31.92</td>
<td>34.48</td>
<td>34.56</td>
</tr>
<tr>
<td>Encoding time (sec)</td>
<td>1959.355</td>
<td>1639.257</td>
<td>1718.904</td>
</tr>
<tr>
<td>Compression Ratio</td>
<td>0.7712</td>
<td>0.7722</td>
<td>0.7722</td>
</tr>
<tr>
<td>Average bits/frame</td>
<td>20766232</td>
<td>20704344</td>
<td>20706376</td>
</tr>
</tbody>
</table>

### Table 3.7 Performance comparisons of DCT-based, DWT-based and Fractal-based transformations for “Mother-daughter” video sequence

<table>
<thead>
<tr>
<th>Parameters</th>
<th>DCT</th>
<th>DWT</th>
<th>Fractal</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSNR Y (dB)</td>
<td>34.48</td>
<td>36.76</td>
<td>36.79</td>
</tr>
<tr>
<td>PSNR U (dB)</td>
<td>34.13</td>
<td>36.19</td>
<td>36.23</td>
</tr>
<tr>
<td>PSNR V (dB)</td>
<td>34.18</td>
<td>36.23</td>
<td>36.27</td>
</tr>
<tr>
<td>Encoding time (sec)</td>
<td>1554.032</td>
<td>1183.067</td>
<td>1182.283</td>
</tr>
<tr>
<td>Compression Ratio</td>
<td>0.9135</td>
<td>0.9135</td>
<td>0.9142</td>
</tr>
<tr>
<td>Average bits/frame</td>
<td>7851616</td>
<td>7766600</td>
<td>7766552</td>
</tr>
</tbody>
</table>

### Table 3.8 Performance comparisons of DCT-based, DWT-based and Fractal-based transformations for “Susie” video sequence

<table>
<thead>
<tr>
<th>Parameters</th>
<th>DCT</th>
<th>DWT</th>
<th>Fractal</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSNR Y (dB)</td>
<td>34.37</td>
<td>36.30</td>
<td>36.34</td>
</tr>
<tr>
<td>PSNR U (dB)</td>
<td>34.44</td>
<td>36.49</td>
<td>36.54</td>
</tr>
<tr>
<td>PSNR V (dB)</td>
<td>34.45</td>
<td>36.47</td>
<td>36.52</td>
</tr>
<tr>
<td>Encoding time (sec)</td>
<td>651.373</td>
<td>581.145</td>
<td>593.079</td>
</tr>
<tr>
<td>Compression Ratio</td>
<td>0.9237</td>
<td>0.9246</td>
<td>0.9244</td>
</tr>
<tr>
<td>Average bits/frame</td>
<td>3475376</td>
<td>3440328</td>
<td>3446912</td>
</tr>
</tbody>
</table>
3.6 Conclusion

The results presented here are comparable and better than the other coding schemes. Most current video coding schemes use motion compensation. Here the fractal techniques used in image coding is used for the video, but in video coding, the advantage of the similarity between the frames are taken and, because of that, higher compression rates are obtained compared with image compression. Experimental results show that this scheme provides a superior performance in terms of PSNR as well as the subjective quality at low bit-rates. Moreover, the blocking artifacts are reduced significantly, the visual quality of the reconstructed frames is better as compared to the other compression methods.