CHAPTER - 2
VIDEO CODEC

Video Compression is a reversible process in which encoding of data is done which contains fewer bits. Through this process we can efficiently store and transmit the data. The inverse process by which we can retrieve the original video is called decompression (decoding). Software and/or hardware used to encode and decode video are called codec. So in short a video codec is a combination of encoder at transmitter side and decoder at receiver side. In this chapter, we will understand MPEG video encoder in detail [16].

2.1 Video Encoder:

The video encoder for MPEG coding technique is as shown below:

![Fig. 2.1: Block diagram of MPEG Video encoder](image1)

The encoding of the digital video is consisting of following steps:

- The first step in the encoding process is to convert the whole digital video in frames, as any digital video is sequence of frames, and arrange them in Group of Picture (GOP) of ten frames to get frame sequence of IBBPBBPBBI.
The second step is to convert all frames into YC\textsubscript{b}C\textsubscript{r} format from RGB format. Here the color space conversion is performed because as per the human visual system, human eye is having lesser sensitivity towards chrominance (color) component compared to luminance (intensity) component, so we can compress chrominance component more and luminance component should be compressed less to keep the quality of video better. Whereas in case of RGB format all of the color components are having equal weightage so we cannot compress all of them in more or less amount. So to get more compression ratio color space conversion is necessary.

Now next step in the encoding process is to get motion compensated video frames. Here the frames are divided in macro blocks and motion estimation is done through which we can get relative position of each macro block, with respect to same block in previous or next frame, which is assigned by motion vectors for each block. According to the motion vectors the frame is predicted and the residue is found out between the original frame and predicted frame, which is known as motion compensated frame.

The fourth step in the encoding process is to take transformation of the residual frames, which converts the residual frame from spatial domain pixel values to frequency domain coefficients with same size.

The next step in the encoding process is quantization of transform coefficients of the frame. The quantization process is done in the compression because the high frequency coefficients of the transform are having lesser information so using quantization process the high frequency coefficients are made zero.

Then in the next step the transform coefficients of frame are scanned in zig-zag manner block wise. Here the coefficients are scanned in zig-zag manner to arrange the coefficients in the ascending order of frequency, as low frequency coefficients are more important. Also another reason for this process is to arrange coefficients in such manner that sequence of zeros is achieved.

At last in the process of encoding the zig-zag scanned coefficients are Run Length Coded to reduce the number of coefficients to be transmitted. The Run Length Coding scheme is subtype of Huffman Coding technique to reduce the data rate for the transmission.
Now let us understand process of each step of encoding video in detail with theoretical concepts and practical implementation of each step on one frame.

### 2.1.1 Color space conversion and Chroma re-sampling:

Any color can be generated in terms of combination the three primary colors: red, blue, and green (RGB). The RGB color space system is one of the ways of representation of color images. Also the luminance (brightness) and chrominance (color) information of any color image can be represented separately. Luminance component of any color image can be represented by weighted sum of the three colors R, G, and B, which represents the “brightness” of the color image. We can also calculate color difference signals $C_r$, $C_b$, and $C_g$, out of which only two are necessary for representation of any color image with Y component, by subtracting the luminance component from each primary color component using the following equations:

\[
Y = 16 + 65.481 \times R + 128.553 \times G + 24.966 \times B \tag{2.1}
\]

\[
C_b = 128 - 37.797 \times R - 74.203 \times G + 112 \times B \tag{2.2}
\]

\[
C_r = 128 + 112 \times R - 93.786 \times G - 18.214 \times B \tag{2.3}
\]

Also from $Y$, $C_b$ and $C_r$ we can calculate R, G and B component for the inverse color space conversion.

\[
R = Y + 1.402 \times (C_r - 128) \tag{2.4}
\]

\[
G = Y - 0.34414 \times (C_b - 128) - 0.71414 \times (C_r - 128) \tag{2.5}
\]

\[
B = Y + 1.772 \times (C_b - 128) \tag{2.6}
\]

Of the three color difference signals, only two are linearly independent of each other, the third one can be expressed in terms of the other two.
The human eye is a less sensitive towards color information than luminance. A conversion of the frames from RGB to YC_bC_r color components leads to use this effect for compression. The chrominance components C_b and C_r can be reduced by down-sampling to 4:2:2, which takes only half of the pixel values in horizontal direction, or to 4:2:0, which takes half of the pixel values in both the directions: horizontal and vertical. The chroma sub-sampling reduces the data volume by 33% in case of 4:2:2 and 50% in case of 4:2:0, as per below equations.

\[
|Y| = |U| = |V|
\]  \hspace{1cm} ----- 2.7

\[
4:2:0 = \frac{|Y| + \frac{1}{4}|U| + \frac{1}{4}|V|}{|Y| + |U| + |V|} = \frac{1}{2}
\]  \hspace{1cm} ----- 2.8

\[
4:2:2 = \frac{|Y| + \frac{1}{2}|U| + \frac{1}{2}|V|}{|Y| + |U| + |V|} = \frac{2}{3}
\]  \hspace{1cm} ----- 2.9

Thus chroma sub-sampling itself gives some amount of compression in the amount of video data.
2.1.2 Motion Estimation and Compensation:

A video can be considered as a sequence of still images known as frames. As two successive frames of a video sequence are taken with small time delay they will be having small changes in pixel values, this temporal redundancy can be removed by motion estimation and compensation algorithms. There are three types of frames in each group of picture [30]:

- I-frames (Intra-coded)
- P-frames (Predicted)
- B-frames (Bi-directionally predicted).

I-frames in each group of picture play key role, they act as reference to other frames so not having reference to other frames and their compression cannot be much higher. I-frame is the reference frame which is transmitted at the first and last position of the GOP, which is not having any motion estimation or compensation implemented. This frame is transmitted as it is to have reference for other frames.

P-frames are being predicted either from an earlier I-frame or from P-frame. As they are predicted from other frame, P-frames can be reconstructed using the reference frame only, as only difference between reference frame and predicted one is transmitted, P-frames need less space than the I-frames. The P-frames are located at fourth and seventh position in the GOP[30].

![Fig. 2.4: Prediction of P and B frames](image-url)
B-frames can be predicted bi-directionally: one forward direction i.e. from upcoming I or P-frame and one backward direction i.e. from past I or P-frame. The position of B-frame is second, third, fifth, sixth, eighth and ninth in the GOP.

P-frames and B-frames are called inter coded frames, whereas I-frames are known as intra coded frames.

The quality of the decoded video and the compression ratio achieved by video codec is dependent on the selection of the sequence of the frames in the group of picture. The reference frame I is going to maintain the quality of the video good but will not be able to give higher compression ratio, whereas P and B type of frames are giving the higher compression ratio but reducing the quality of the decoded video. In practice the tradeoff between these two must be maintained for the better performance of the video codec. The normally used frame sequence in standard group of picture is: **IBBPBBPBBP**BBP…[30].

![Group of Picture (GOP)](image)

Fig. 2.5: Group of Picture (GOP)

The similarities between the frames in group of picture found out using the *motion estimation or motion compensation* process. The motion vector shows the similarity between two frames. The resultant frame quality highly depends on the efficiency of the motion estimation and compensation algorithm. For higher compression ratio and the best quality decoded video the motion estimation and compensation algorithm
should be good enough. Though, motion estimation gives us better compression in the encoded video, it is a computationally intense operation, which is not suitable for all type of real time applications. The steps followed by motion estimation algorithm are as follows:

Frame Segmentation - The frame segmentation is process of dividing frames into macro blocks usually of the size 8 x 8 or 16 x 16 pixels. In selecting the block size one must keep in mind two things: if smaller block size is taken, more motion vectors are needed; and if the block size is taken too large, matching of macro blocks will be less correlated. Usually block sizes of 16x16 pixels is being used by MPEG.

Search Threshold - It is the level of similarity checked for the comparison of two blocks of frames located at same place. Search threshold is the level which is used to decide whether the difference between two blocks is to be transmitted of the whole block.

Block Matching – The block matching does the process of arranging the macro blocks of the previous frame to generate an actual predicted frame. The most time consuming process in the encoding is block matching process. During the block matching process the luminance part of the frame is used and each block of the current frame is compared with all the blocks of the previous frame. The critical factor for selection of the block matching is search area. If the search area is larger the block found out by the motion estimation will be most likely. So it is observed that the number of search operations increases, as the search area is increased. So that larger search areas are not selected as due to that the encoding process is slowing down. The solution for this problem is to select rectangular shaped macro blocks which take into account horizontal as well as vertical movement of the picture [3] [4].
**Prediction Error Coding** – Normally in the MPEG encoding process, from the motion estimation algorithm the next frame is predicted and then residual is found out between the original frame and the predicted one [30]. The residual found out is called as prediction error and then the prediction error is being coded for the transmission purpose. The figure 2.7 shows the process.

![Fig. 2.6: Block Matching Process](image)

![Fig. 2.7: Motion estimation](image)

**Vector Coding** – As majority of the group of picture is containing B and P frames, these frames contain motion vectors to be stored or transmitted with it. So these motion vectors are also to be compressed to reduce the number of bits used to represent them. These motion vectors are having high correlation between them, so it is easy to compress them [5].
2.1.3 Transform coding:

The motion compensated residual frames are ready for transmission now. So to convert it in frequency domain normal MPEG takes Discrete Cosine Transform (DCT) of the frames [29] [30]. Cosine Transform is an even (real) part of Fourier Transform, so it is having almost similar properties as Fourier Transform. The process gives the representation of the frame in terms of the increasing frequency coefficients block wise which are magnitudes of cosine functions, i.e. real numbers in nature.

Representation of the frame using DCT is just like Fast Fourier Transform (FFT). So the macro blocks (8x8 or 16x16 pixels) are being considered as frequency components. Here considering fixed macro block size puts limitation on the compression efficiency, but due to level of computational complexity and easy implementation we cannot take variable macro block size.

The Discrete Cosine Transform is described by the following formula:

\[
F(u, v) = \frac{2}{N} C(u) C(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y) \cos \left( \frac{(2x+1)\pi u}{2N} \right) \cos \left( \frac{(2y+1)\pi v}{2N} \right) \quad \text{2.10}
\]

Where, \( C(u), C(v) = 1/\sqrt{2} \) for \( u, v = 0 \)

\[ C(u), C(v) = 1 \text{ for } u, v = \text{else} \]
The DCT is computationally very expensive and its complexity increases as the size of frame increases. That’s why only the video compression algorithms divide the frames into macro blocks before DCT.

And inverse DCT can be described by the following formula:

\[
f(x, y) = \frac{2}{N} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} C(u)C(v)F(u, v) \cos \frac{(2x+1)\pi}{2N} \cos \frac{(2y+1)\pi}{2N} \]

\[\text{----- 2.11}\]
If we interpret the figure 2.9, then the top left most term gives the constant value which is representation of the average gray level of the image. Other remaining terms represent the details of the image. As we move from top to bottom or left to right in the DCT of frame the frequency of the cosine function increases. Up to now we have seen many steps but are not reducing the redundancy in the frame, but just arranging the frame in a proper way to compress.

2.1.4 Quantisation and Zig-zag scanning:

In the process of quantisation, the DCT coefficients of the frames are divided by the quantisation matrix block wise. Here the quantisation matrix is defined such that it reduces the high frequency terms of the transform domain representation to zero, as the human perception says that human eyes are less sensitive towards high frequency components which are minute details of the frame in spatial domain [29] [30].

\[
F_{QUANTIZED} = \frac{F(u,v)}{Q(u,v)} \quad \text{----- 2.12}
\]

Where, Q is the quantisation Matrix of dimension N x N. The quality of the frame degrades due to quantisation but can be controlled by choosing the quantisation matrix. The standard quantization matrix for MPEG codecs is as given below:
One example of quantisation is as shown below, which shows one 8 x 8 block of original image coefficients and values of the same coefficient block after quantisation.

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>11</td>
<td>10</td>
<td>16</td>
<td>29</td>
<td>40</td>
<td>53</td>
<td>68</td>
</tr>
<tr>
<td>12</td>
<td>12</td>
<td>16</td>
<td>23</td>
<td>32</td>
<td>43</td>
<td>56</td>
<td>71</td>
</tr>
<tr>
<td>14</td>
<td>13</td>
<td>21</td>
<td>28</td>
<td>37</td>
<td>48</td>
<td>61</td>
<td>76</td>
</tr>
<tr>
<td>20</td>
<td>23</td>
<td>28</td>
<td>35</td>
<td>44</td>
<td>55</td>
<td>58</td>
<td>83</td>
</tr>
<tr>
<td>29</td>
<td>32</td>
<td>37</td>
<td>44</td>
<td>53</td>
<td>64</td>
<td>77</td>
<td>92</td>
</tr>
<tr>
<td>40</td>
<td>43</td>
<td>48</td>
<td>55</td>
<td>64</td>
<td>75</td>
<td>88</td>
<td>103</td>
</tr>
<tr>
<td>53</td>
<td>56</td>
<td>61</td>
<td>68</td>
<td>77</td>
<td>88</td>
<td>101</td>
<td>116</td>
</tr>
<tr>
<td>68</td>
<td>71</td>
<td>76</td>
<td>83</td>
<td>92</td>
<td>103</td>
<td>116</td>
<td>131</td>
</tr>
</tbody>
</table>

If after the quantisation process more number of zeros are there in the coefficients, the frame degradation will be more and more amount of blocking artifacts will be visible in the decoded frames. This will give higher compression as the blocks are compressed individually, but when new blocks are present in next frame more amount of blocking artifacts will be seen.

Then in the next step, the DCT coefficients of frame are scanned in zig-zag manner, as shown in the Fig. 2.10, block wise. Here the coefficients are scanned in zig-zag manner to arrange the coefficients in the ascending order of frequency, as low frequency coefficients are more important. Also another reason for this process is to arrange coefficients in such manner that sequence of zeros is achieved[30].
2.1.5 Run Length Coding and Huffman Coding (Entropy Coding):

Video compression techniques generally use two level Entropy coding scheme which contain first step of Run length coding and second stage of Variable length coding as a Huffman coding scheme [14].

In the first step, the consecutive repeating numbers are represented by only two numbers. Taking one example, if we receive (5,8) as RLC coded numbers in this first number represents run i.e. how many times the number is repeating and second number shows the number to be repeated, then it is decoded as (8,8,8,8,8) and sequence of the numbers [1].

The Huffman coding techniques uses the standard coding table for the video compression algorithms defined by MPEG, which contains bit representation of all the numbers occurring. The numbers which are occurring often are represented by less number of bits and less frequent numbers are represented by more number of bits.

As discussed previously, frequently occurring RLC patterns are coded with the least number of bits. After this the stream of digital data, which represents the picture, is ready to be stored on the optical disk or can be prepared for the transmission on the channel.
2.2  Video Decoder:

![Block diagram of MPEG video decoder](image)

Fig. 2.12: Block diagram of MPEG video decoder

At the receiver end exactly inverse steps are performed, than transmitter end.

- As the receiver receives the MPEG bit stream the data bits are run length decoded and the quantized coefficients of DCT are achieved.
- Then these coefficients are de-quantized using the same quantization matrix and DCT coefficients are recovered.
- Inverse DCT of these coefficients is taken, which leads to the motion compensated frames.
- Then from these GOP the original frames of video are achieved using motion compensation and estimation algorithms.
- Then chrominance components are re-sampled and frames are converted back to RGB format from YCbCr format.
2.3 Results:

Here the MPEG-2 algorithm is implemented in MATLAB version 7.10.0.499a, as per the steps explained above in the video encoder and decoder sections. For testing purpose five different standard videos are taken which are having differences in the motion, color component etc. from each other and having size of 128 x 128 pixels to make study uniform.

Then for the measuring all parameters one popular test bench named, MSU Video Quality Measurement Tool 3.0 has been used. This tool is measuring all the parameters mentioned in the chapter-1, for the quality measurement of the decoded video from its compressed counterpart.

Figure 2.13 shows compressed video frames of the digital video named, ‘foreman.avi’. According to frame rate 30 frames/second, 1 second video will contain 30 frames, as shown in the figure 2.13.

The compression ratio for the video ‘foreman.avi’ is as below:

\[
Compression \ ratio = \frac{Compressed \ Video \ Data}{Original \ Video \ Data}
\]

So from this equation,

\[
Compression \ ratio = \frac{(21856 + 23367 + 21869) * 2 + 73728 + 54}{128 * 128 * 3 * 30} = 0.114
\]

The above calculation is including the following details: the final coefficients to be transmitted after run length encoding, coefficients of motion vector and flag bits for bi-directionally predicted frames in the numerator and in denominator number of pixels in 1 second color video with size of 128 x 128.
Now, the average values of subjective quality measurement parameters are as shown in below table:

Table 2.1: Average values of parameters for ‘forman.avi’

<table>
<thead>
<tr>
<th>PSNR (dB)</th>
<th>MSE</th>
<th>SSIM</th>
<th>MSAD</th>
<th>VQM</th>
<th>Blocking Beta Original</th>
<th>Blocking Beta Compressed</th>
<th>Blurring Beta Original</th>
<th>Blurring Beta Compressed</th>
</tr>
</thead>
<tbody>
<tr>
<td>22.65</td>
<td>353.47</td>
<td>0.7366</td>
<td>10.787</td>
<td>4.5864</td>
<td>7.15</td>
<td>7.914</td>
<td>24.52</td>
<td>26.808</td>
</tr>
</tbody>
</table>

All the above parameters are averaged over 30 frames, and their frame wise values are as shown in the figure 2.14.
(a) Peak Signal to Noise Ratio

(b) Mean Square Error
(c) Structural Similarity

(d) Mean Sum of Absolute Difference
(e) Video Quality Matrices

(f) Blocking Matrices
Figure 2.15 shows compressed video frames of the digital video named, ‘vipmen.avi’. According to frame rate 30 frames/second, 1 second video will contain 30 frames, as shown in the figure 2.15.

The compression ratio for the video ‘vipmen.avi’ is as below:

\[
Compression\ ratio = \frac{Compressed\ Video\ Data}{Original\ Video\ Data}
\]

So from this equation,

\[
Compression\ ratio = \frac{((12158 + 12803 + 15429) \times 2) + 73728 + 54}{128 \times 128 \times 3 \times 30} = 0.0867
\]

The above calculation is including the following details: the final coefficients to be transmitted after run length encoding, coefficients of motion vector and flag bits for bi-directionally predicted frames in the numerator and in denominator number of pixels in 1 second color video with size of 128 x 128.
Now, the average values of subjective quality measurement parameters are as shown in below table:

Table 2.2: Average values of parameters for ‘vipmen.avi’

<table>
<thead>
<tr>
<th>PSNR (dB)</th>
<th>MSE</th>
<th>SSIM</th>
<th>MSAD</th>
<th>VQM</th>
<th>Blocking Beta</th>
<th>Blurring Beta</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Original</td>
<td>Original</td>
</tr>
<tr>
<td>21.58</td>
<td>451.54</td>
<td>0.8139</td>
<td>9.124</td>
<td>5.3778</td>
<td>15.007</td>
<td>13.7187</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Compressed</td>
<td>Compressed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>15.757</td>
<td>18.577</td>
</tr>
</tbody>
</table>

All the above parameters are averaged over 30 frames, and their frame wise values are as shown in the figure 2.16.
(a) Peak Signal to Noise Ratio

(b) Mean Square Error
(c) Structural Similarity

(d) Mean Sum of Absolute Difference
(e) Video Quality Matrices

(f) Blocking Matrices
Figure 2.17 shows compressed video frames of the digital video named, ‘viplane.avi’. According to frame rate 30 frames/second, 1 second video will contain 30 frames, as shown in the figure 2.17.

The compression ratio for the video ‘viplane.avi’ is as below:

\[
\text{Compression ratio} = \frac{\text{Compressed Video Data}}{\text{Original Video Data}}
\]

So from this equation,

\[
\text{Compression ratio} = \frac{(16734 + 18753 + 17069) \times 2 + 73728 + 54}{128 \times 128 \times 3 \times 30} = 0.103
\]

The above calculation is including the following details: the final coefficients to be transmitted after run length encoding, coefficients of motion vector and flag bits for bi-directionally predicted frames in the numerator and in denominator number of pixels in 1 second color video with size of 128 x 128.
Now, the average values of subjective quality measurement parameters are as shown in below table:

Table 2.3: Average values of parameters for ‘viplane.avi’

<table>
<thead>
<tr>
<th>PSNR (dB)</th>
<th>MSE</th>
<th>SSIM</th>
<th>MSAD</th>
<th>VQM</th>
<th>Blocking Beta</th>
<th>Blurring Beta</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Original</td>
<td>Compressed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Original</td>
<td>Compressed</td>
</tr>
<tr>
<td>20.245</td>
<td>614.57</td>
<td>0.6247</td>
<td>12.007</td>
<td>6.54</td>
<td>12.673</td>
<td>11.9655</td>
</tr>
<tr>
<td></td>
<td>14.81</td>
<td>17.3725</td>
<td></td>
<td></td>
<td>14.81</td>
<td>17.3725</td>
</tr>
</tbody>
</table>

All the above parameters are averaged over 30 frames, and their frame wise values are as shown in the figure 2.18.
(a) Peak Signal to Noise Ratio

(b) Mean Square Error
(c) Structural Similarity

(d) Mean Sum of Absolute Difference
(e) Video Quality Matrices

(f) Blocking Matrices
Figure 2.19 shows compressed video frames of the digital video named, ‘viplanedeparture.avi’. According to frame rate 30 frames/second, 1 second video will contain 30 frames, as shown in the figure 2.19.

The compression ratio for the video ‘viplanedeparture.avi’ is as below:

\[
\text{Compression ratio} = \frac{\text{Compressed Video Data}}{\text{Original Video Data}}
\]

So from this equation,

\[
\text{Compression ratio} = \frac{\left(17057 + 15156 + 14592 \right) + 73728 + 54}{128 \times 128 \times 3 \times 30} = 0.0861
\]

The above calculation is including the following details: the final coefficients to be transmitted after run length encoding, coefficients of motion vector and flag bits for bi-directionally predicted frames in the numerator and in denominator number of pixels in 1 second color video with size of 128 x 128.

(g) Blurring Matrices
Now, the average values of subjective quality measurement parameters are as shown in below table:

<table>
<thead>
<tr>
<th>Table 2.4: Average values of parameters for ‘viplancedeparture.avi’</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PSNR (dB)</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Original</td>
</tr>
<tr>
<td>23.54</td>
</tr>
</tbody>
</table>
(a) Peak Signal to Noise Ratio

(b) Mean Square Error
(c) Structural Similarity

(d) Mean Sum of Absolute Difference
(e) Video Quality Matrices

(f) Blocking Matrices
Figure 2.21 shows compressed video frames of the digital video named, ‘viptraffic.avi’. According to frame rate 30 frames/second, 1 second video will contain 30 frames, as shown in the figure 2.21.

The compression ratio for the video ‘viptraffic.avi’ is as below:

\[
\text{Compression ratio} = \frac{\text{Compressed Video Data}}{\text{Original Video Data}}
\]

So from this equation,

\[
\text{Compression ratio} = \frac{((13147 + 14909 + 19944) \times 2) + 73728 + 54}{128 \times 128 \times 3 \times 30} = 0.101
\]

The above calculation is including the following details: the final coefficients to be transmitted after run length encoding, coefficients of motion vector and flag bits for bi-directionally predicted frames in the numerator and in denominator number of pixels in 1 second color video with size of 128 x 128.
Now, the average values of subjective quality measurement parameters are as shown in below table:

<table>
<thead>
<tr>
<th>PSNR (dB)</th>
<th>MSE</th>
<th>SSIM</th>
<th>MSAD</th>
<th>VQM</th>
<th>Blocking Beta</th>
<th>Blurring Beta</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Original</td>
<td>Compressed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Original</td>
<td>Compressed</td>
</tr>
<tr>
<td>18.77</td>
<td>862.67</td>
<td>0.6688</td>
<td>16.127</td>
<td>6.69</td>
<td>13.09</td>
<td>12.58</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>21.32</td>
<td>23.34</td>
</tr>
</tbody>
</table>

All the above parameters are averaged over 30 frames, and their frame wise values are as shown in the figure 2.22.
(a) Peak Signal to Noise Ratio

(b) Mean Square Error
(c) Structural Similarity

(d) Mean Sum of Absolute Difference
(e) Video Quality Matrices

(f) Blocking Matrices
2.4 Conclusion:

In this chapter, we have discussed normal MPEG video codec and implemented it in MATLAB. The results of the compressed video for MPEG video codec shows average video compression ratio in the range of 9 to 12%. This amount of video compression is sufficient for some applications like TV broadcasting over cable TV or satellite TV and DVD quality video storage and playback. But if we think about the transferring the same video over internet through PSTN or mobile networks, it is necessary to have higher compression ratio, as these networks are having limited bandwidth compared to TV broadcasting networks. So to achieve higher compression ratio, the MPEG video codec must be enhanced.

While thinking of enhancement in the MPEG video codec, one must think about the complexity of algorithm, as the necessity of higher compression ratio is for the applications like internet and mobile communication, as the devices are compact and having limitations in processor and memory of the devices. Keeping this thing in mind I have implemented one modification in MPEG video codec, which is not much complex in nature, though giving higher compression ratio.