International Conference on Intelligent Technologies & Science - 2021
(ICITS-2021)

An Advanced Video Compression Method using Accordion DCT Scheme

Pragya Singh, Yogendra Pratap Singh
Computer Science and Engineering,
Goel Institute Of Technology and Management, Lucknow
pragya2416@gmail.com, yogendra.p.simgh@goel.edu.in

Abstract: Video compression technique is now mature as is proven by the large number of applications that make use of DWT and DCT technology. Now day's lot of video compression techniques proposed. With efficient compression techniques, a significant reduction in file size can be achieved with little or no adverse effect on the visual quality. This paper gives the idea about for video compression technique but not very much good for the real time video compression techniques either have a demerit of loosely techniques like DCT and DWT but here we are going to present a noble technique in which we will use object position change finding algorithm to get our video process in real time and having lossless decompressions. Compression is done in real time, such a way while maintaining the benefits of keeping all of the information of the source and also the benefits of compression during the production process. "Lossless" means that the output from the decompressor is bitfor-bit identical with the original input to the compressor. The decompressed video stream should be completely identical to original. In addition to providing improved coding efficiency in real time the technique provides the ability to selectively encode, decode, and manipulate individual objects in a video stream. The technique used results in video coding that a high compression ratio can be obtained without any loss in data in real time.

Keyword: Discrete Cosine Transform (DCT), Discrete Wavelet Transform (DWT), MPEG, Video Coding.

1. Introduction:

Over the past decades, video compression technologies have become an integral part of the way we create, communicate and consume visual information. Digital video communication can be found today in many applications such as broadcast services over satellite and terrestrial channels, digital video storage, wires and wireless conversational services and etc.
The data quantity is very large for the digital video and the memory of the storage devices and the bandwidth of the transmission channel are not infinite, so reducing the amount of data needed to reproduce video saves storage space, increases access speed and is the only way to achieve motion video on digital computers.
For instance, we have a 720 x 480 pixels per frame, 30 frames per second, total 90 minutes full color video, then the full data quantity of this video is about 167.96 G bytes. This raw video contains an immense amount of data, and communication and storage capabilities are limited and expensive. Thus, several video compression algorithms had been developed to reduce the data quantity and provide the acceptable quality as possible as they can. This tutorial starts with an explanation of the basic concepts of video compression algorithms and then introduces two international standards, known as MPEG-1 and MPEG-2.
Why can video be compressed? The reason is that video contains much spatial and temporal redundancy. In a single frame, nearby pixels are often correlated with each other. This is called spatial redundancy, or the intraframe correlation. Another one is temporal redundancy, which means adjacent frames are highly correlated, or called the interframe correlation. Therefore, our goal is to efficiently reduce spatial and temporal redundancy to achieve video compression.

2. The Rise of Efficient Compression:

The carousel of progress keeps on turning, and today's compression algorithms are much more effective than older ones. The Cinepak codec that powered early versions of both QuickTime and Windows Media video formats aimed no higher than getting 320x240 video resolution out of a standard CD-ROM drive, which means cramming 2.2Mbps plus audio and overhead into a 1.2Mbps traffic stream. Call it 50 percent compression on a good day.
If you never saw a digital video back in the 1990s, you're not alone. The files were large and the downloads were slow. And while the tiny hard drives of the era would have welcomed some video compression with open arms, the other parts of that equation were simply not there; an Intel 486 or Pentium would grind to a standstill trying to make sense of a simplistic Motion JPEG or MPEG-1 video, even at very low resolutions. But better days were just around the corner.
The popular MPEG-2 standard pretty much destroyed Cinepak, Intel Indeo, and other early codecs in the late 1990s by compressing video streams to as much as 1/30 of the original video size while still maintaining acceptable picture quality. That's hefty enough to let that old 1x CD-ROM handle a full, standard-resolution NTSC signal at about 1Mbps. This is the format used in DVD video, many digital broadcasts, and most online video streams.
MPEG-2 can present a 1080p video in a 2Mbps envelope, but with horrible blocking artifacts from all the compression if you take it that far. These codecs are lossy, which means that

you must always balance image quality against file size. HD video was not exactly what MPEG-2 was made for.
By the standards of modern hardware, like recent models of Texas Instruments' OMAP processors and anything from Intel or AMD, MPEG-2 encoding is a walk in the park. Newer formats, not so much. This is why TiVo box still stores DVR recordings in this age-old format.

3. Lossless and Lossy Compression Algorithms:

Compression is used just about everywhere. All the images we get on the web are compressed, typically in the JPEG or GIF formats, most modems use compression, HDTV will be compressed
using MPEG-2, and several file systems automatically compress files when stored, and the rest of us do it by hand. The neat thing about compression, as with the other topics we will cover in this course, is that the algorithms used in the real world make heavy use of a wide set of algorithmic tools, including sorting, hash tables, tries, and FFTs. Furthermore, algorithms with strong theoretical foundations play a critical role in real-world applications.
The task of compression consists of two components, an encoding algorithm that takes a message and generates a “compressed” representation (hopefully with fewer bits), and a decoding algorithm that reconstructs the original message or some approximation of it from the compressed representation. These two components are typically intricately tied together since they both have to understand the shared compressed representation.
We distinguish between lossless algorithms, which can reconstruct the original message exactly from the compressed message, and lossy algorithms, which can only reconstruct an approximation of the original message. Lossless algorithms are typically used for text, and lossy for images and sound where a little bit of loss in resolution is often undetectable, or at least acceptable. Lossy is used in an abstract sense, however, and does not mean random lost pixels, but instead means loss of a quantity such as a frequency component, or perhaps loss of noise. For example, one might think that lossy text compression would be unacceptable because they are imagining missing or switched characters. Consider instead a system that reworded sentences into a more standard form, or replaced words with synonyms so that the file can be better compressed. Technically the compression would be lossy since the text has changed, but the “meaning” and clarity of the message might be fully maintained, or even improved. In fact Strunk and White might argue that good writing is the art of lossy text compression.
Is there a lossless algorithm that can compress all messages? There has been at least one patent application that claimed to be able to compress all files (messages)—Patent 5,533,051 titled “Methods for Data Compression”. The patent application claimed that if it was applied recursively, a file could be reduced to almost nothing. With a little thought we should convince ourself that this is not possible, at least if the source messages can contain any bit-sequence. We can see this by a simple counting argument. Lets consider all 1000 bit messages, as an example. There are 21000 different messages we can send, each which needs to be distinctly identified by the decoder. It should be clear we can’t represent that many different messages by sending 999 or fewer bits for all the messages — 999 bits would only allow us to send 2999 distinct messages. The truth is that if any one message is shortened by an algorithm, then some other message needs to be lengthened. We can verify this in practice by running GZIP on a GIF file. It is, in fact, possible to go further and show that for a set of input messages of fixed length, if one message is compressed, then the average length of the compressed messages over all possible inputs is always going to be longer than the original input messages. Consider, for example, the 8 possible 3 bit messages. If one is compressed to two bits, it is not hard to convince yourself that two messages will have to expand to 4 bits, giving an average of 3 1/8 bits. Unfortunately, the patent was granted.
Because one can’t hope to compress everything, all compression algorithms must assume that there is some bias on the input messages so that some inputs are more likely than others, i.e. that there is some unbalanced probability distribution over the possible messages. Most compression algorithms base this “bias” on the structure of the messages – i.e., an assumption that repeated characters are more likely than random characters, or that large white patches occur in “typical” images. Compression is therefore all about probability.
When discussing compression algorithms it is important to make a distinction between two components: the model and the coder. The model component somehow captures the probability distribution of the messages by knowing or discovering something about the structure of the input.
The coder component then takes advantage of the probability biases generated in the model to generate codes. It does this by effectively lengthening low probability messages and shortening high-probability messages. A model, for example, might have a generic “understanding” of human faces knowing that some “faces” are more likely than others (e.g., a teapot would not be a very likely face). The coder would then be able to send shorter messages for objects that look like faces. This could work well for compressing teleconference calls. The models in most current real-world compression algorithms, however, are not so sophisticated, and use more mundane measures such as repeated patterns in text. Although there are many different ways to design the model component of compression algorithms and a huge range of levels of sophistication, the coder components tend to be quite generic—in current algorithms are almost exclusively based on either Huffman or arithmetic codes. Lest we try to make to fine of a distinction here, it should be pointed out that the line between model and coder components of algorithms is not always well defined.
It turns out that information theory is the glue that ties the model and coder components together. In particular it gives a very nice theory about how probabilities are related to

information content and code length. As we will see, this theory matches practice almost perfectly, and we can achieve code lengths almost identical to what the theory predicts.
Another question about compression algorithms is how does one judge the quality of one versus another. In the case of lossless compression there are several criteria we can think of, the time to compress, the time to reconstruct, the size of the compressed messages, and the generality—i.e., does it only work on Shakespeare or does it do Byron too. In the case of lossy compression the judgment is further complicated since we also have to worry about how good the lossy approximation is. There are typically tradeoffs between the amount of compression, the runtime, and the quality of the reconstruction. Depending on your application one might be more important than another and one would want to pick your algorithm appropriately. Perhaps the best attempt to systematically compare lossless compression algorithms is the Archive Comparison Test (ACT) by Jeff Gilchrist. It reports times and compression ratios for 100s of compression algorithms over many databases. It also gives a score based on a weighted average of runtime and the compression ratio.

4. Result and Discussion:

The simulation on videos comression is performed using matlab tool .We have taken videos from avi files present in matlab. About 5 videos are taken to check the compression ratio and PSNR for both cases using DCT based compression and with ACC DCT based compression.
Before consederirng the results of video we tested the proposed algorithm on single image to demonstrate the steps wise methodology and its uses in compression technique.
For this purpose we have taken a gray scale image after subsampling with each plane must be split into 8×8 blocks. Depending on chroma subsampling, this yields (Minimum Coded Unit) MCU blocks of size 8×8 (4:4:4 – no subsampling), 16×8 (4:2:2), or most commonly 16×16 (4:2:0). In video compression MCUs are called macroblocks.
Discrete cosine transform: Next, each 8×8 block of each component (Y, Cb, Cr) is converted to a frequency-domain representation, using a normalized, two-dimensional type-II discrete cosine transform (DCT) (figure 1).

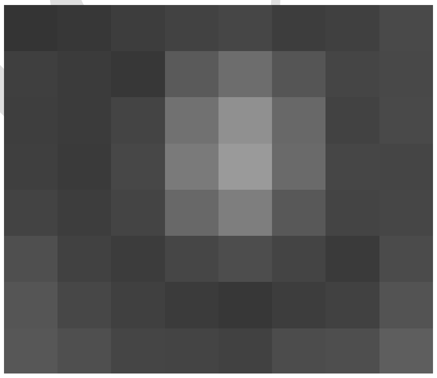
Fig. 1: The 8×8 sub-image shown in 8-bit grayscale

STEP-1:Taking group of pictures from video frames (figure 1) .GOP=10

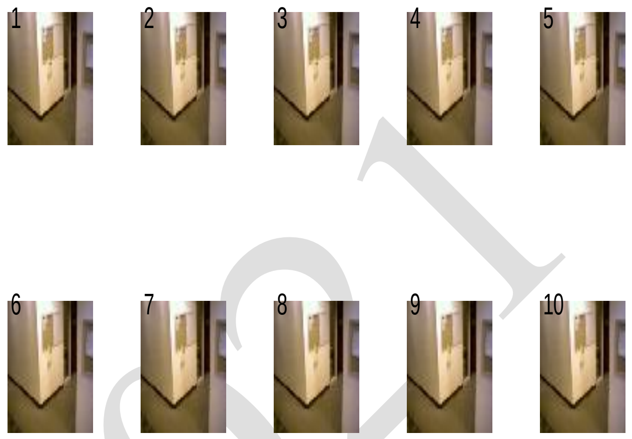

Fig 2: GOP from 1 to 10 from video vipmen.avi

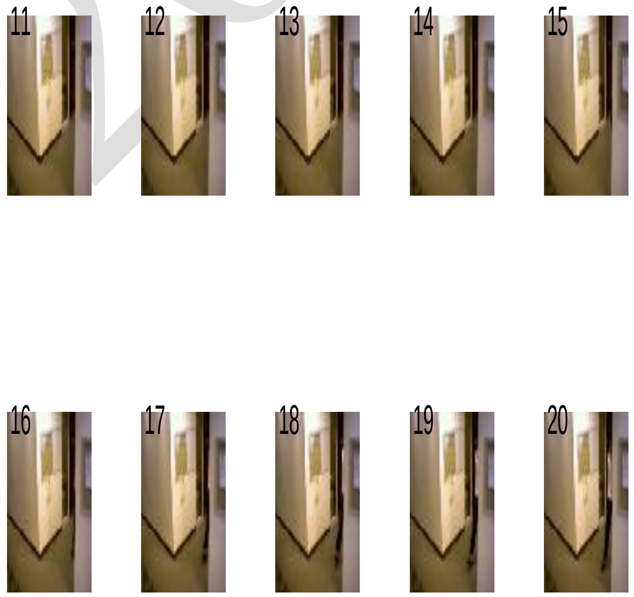

Fig. 3: GOP from 11 to 20 from video vipmen.avi

Similar to fig. 2 we take 10 GOP at each iteration as shown in figure 3 at next iteration GOP from 11 to 20 are taken.

STEP- 2: Construction of accordion image: From each GOP we combine the picture to make a single accordian image prior to compression.An example of how to make accordian image/matrix is given in figure 4.For simplicity we have taken GOP =3 and matrix of 3X3.

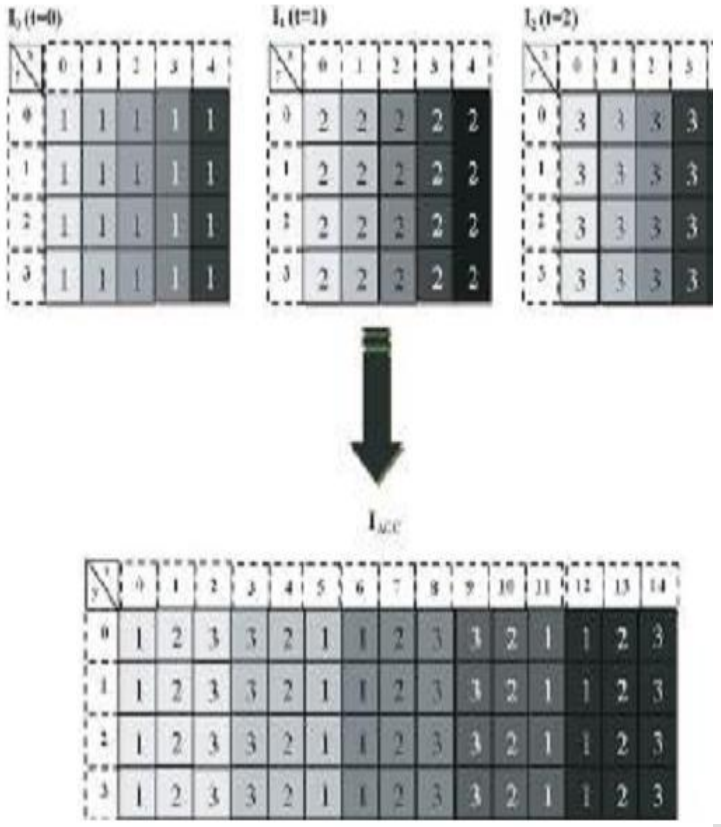

Fig. 4: Accordion Representation Example

Similar to example of figure 4 we have constructed accordion image of GOP =10 .Where each image in a group was of size 120x160x3 after getting the accordion image we get an image of 1200x160x3. One of the accordion image is shown in figure 5. This accordion image is obtained from GOP 11 to 20 (figure 2).

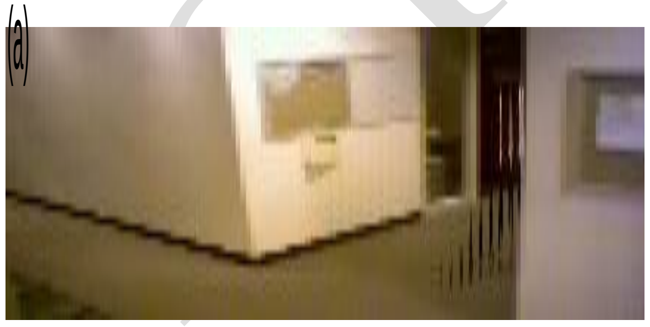

Fig. 5: Accordian Image obtained from GOP 11 to 20.

STEP- 3: Compresion and Decoding: After getting the accordion image we follow the video compression for each GOP. Table 1 shows the results of video compression for each iteration.

TABLE 1

GROUP OF PICTURE	**COMPRESSION RATE**	**PSNR**
1 to 10	1.3326 Bits / pixel	39.0815
11 to 20	1.443 Bits / pixel	38.3896
21 to 30	1.7229 Bits / pixel	36.7183
31 to 40	1.919 Bits / pixel	35.9532
41 to 50	1.7385 Bits / pixel	36.7006
51 to 60	1.5453 Bits / pixel	37.9957
……………	…………….	………………
211 to 220	1.7281 Bits / pixel	36.8345
221 to 230	1.8248 Bits / pixel	36.7684
231 to 240	1.7502 Bits / pixel	36.2355
241 to 250	1.9454 Bits / pixel	34.45
251 to 260	2.7633 Bits / pixel	38.8741
261 to 270	1.3844 Bits / pixel	37.6222

Average PSNR= 37.1403
NET Compression=Original file/Compressed file = 13.8992
Step 4: Decompression and Decoding: After compression we can perform decompression of video to compare the mean square error. Figure 5 shows the accordion image obtained after reconstruction from compressed video.

5. Conclusion:

In this work, we successfully extended and implemented a ACC-DCT based video compression using up/down sampling algorithm on MATLAB and provided experimental results to compare our method with the DCT based compression of existing methods. We not only improved the coding efficiency in the proposed encoding algorithm but also it reduces complexity.
As discussed in the results shown for different videos, proposed method provides benefits of rate-PSNR performance at the good quality of base layer and low quality of enhancement layer. We took 3 videos and performed DCT based and ACC DCT based video compression. The compression ration and PSNR are taken as quality measurement criteria.

References:

[1] Iain E. G. Richardson, H.264 and MPEG-4 Video Compression, Video Coding for Next-generation Multimedia, the Robert Gordon University, Aberdeen, UK, 2003.

[2] Ze-Nian Li and M. S. Drew, “Fundamentals of Multimedia, ” Prentice Hall, 2004.
[3] Yun Q.Shi and Huifang Sun, “Image and Video Compression for Multimedia Engineering: Fundamentals, Algorithms, and Standards”, CRC press, 2000.
[4] Yao Wand, Jorn Ostermann and Ya-Qin Zhang, “Video Processing and Communications”, Prentice Hall, 2007.
[5] Richardson, Lain E. G., “Video Codec Design: Developing Image and Video Compression Systems”, John Wiley & Sons Inc, 2002.
[6] Barry G, Haskell, Atul Puri and Arun N. Netravali, “Digital Video : An Introduction to MPEG-2”, Boston : Kluwer Academic, 1999.
[7] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, “Overview of the H.264/AVC video coding standard”, IEEE Trans. on Circuits and systems for video Technology, vol. 13, no. 7, pp. 560-576, July 2003.
[8] G. Sullivan and T. Wiegand, “Video Compression - From Concepts to theH.264/AVC Standard”, Proceedings of the IEEE, Special Issue on Advances in Video Coding and Delivery, December 2004.
[9] T. Sikora, “MPEG-4 video standard verification model,” IEEE Trans. Circuits Syst. Video Technol., vol. 7, no. 1, pp. 19-31, Feb 1997.
[10] R. Koenen, Editor, “Overview of the MPEG-4 Standard,” ISO/IEC JTC/SC29/WG21, MPEG-99-N2925, March 1999, Seoul, South Korea.
[11] T. Sikora, “MPEG-4 very low bit rate video,” IEEE International Symposium on Circuits and Systems, ISCAS ’97, vol. 2, pp. 1440-1443, 1997.
[12] Y. Ninomiya, Y. Ohtsuka, Y. Izumi, S. Gohshi, and Y. Iwadate, An HDTV broadcasting system utilizing a bandwidth compression technique-MUSE, IEEE Trans Broadcasting 33 (1987), 130–160.
[13] D. Le Gall, MPEG: A video compression standard for multimedia applications, Commun ACM 34 (1991), 46–58.
[14] H. Schwarz, and T. Wiegand, The emerging JVT/H. 26L video coding standard, Proc IBC, 2002.
[15] H. Jung, K. Sung, K.S. Nayak, E.Y. Kim, and J.C. Ye, k-t FOCUSS: A general compressed sensing framework for high resolution dynamic MRI, Magn Reson Med 61 (2009), 103–116.
[16] Z. Liang, and P. Lauterbur, An efficient method for dynamic magnetic resonance imaging, IEEE Trans Med Imaging 13 (1994), 677–686.
[17] H. Jung, J. Park, J. Yoo, and J.C. Ye, Radial k-t FOCUSS for high-resolution cardiac cine MRI, Magn Reson Med 63 (2010), 68–78.
[18] K. Ramchandran, A. Ortega, and M. Vetterli, Bit allocation for dependent quantization with applications to multiresolution and MPEG video coders, IEEE Trans Image Process 3 (1994), 533–545.
[19] B. Girod, A. Aaron, S. Rane, and D. Rebollo-Monedero, Distributed video coding, Proc IEEE 93 (2005), 71–83.

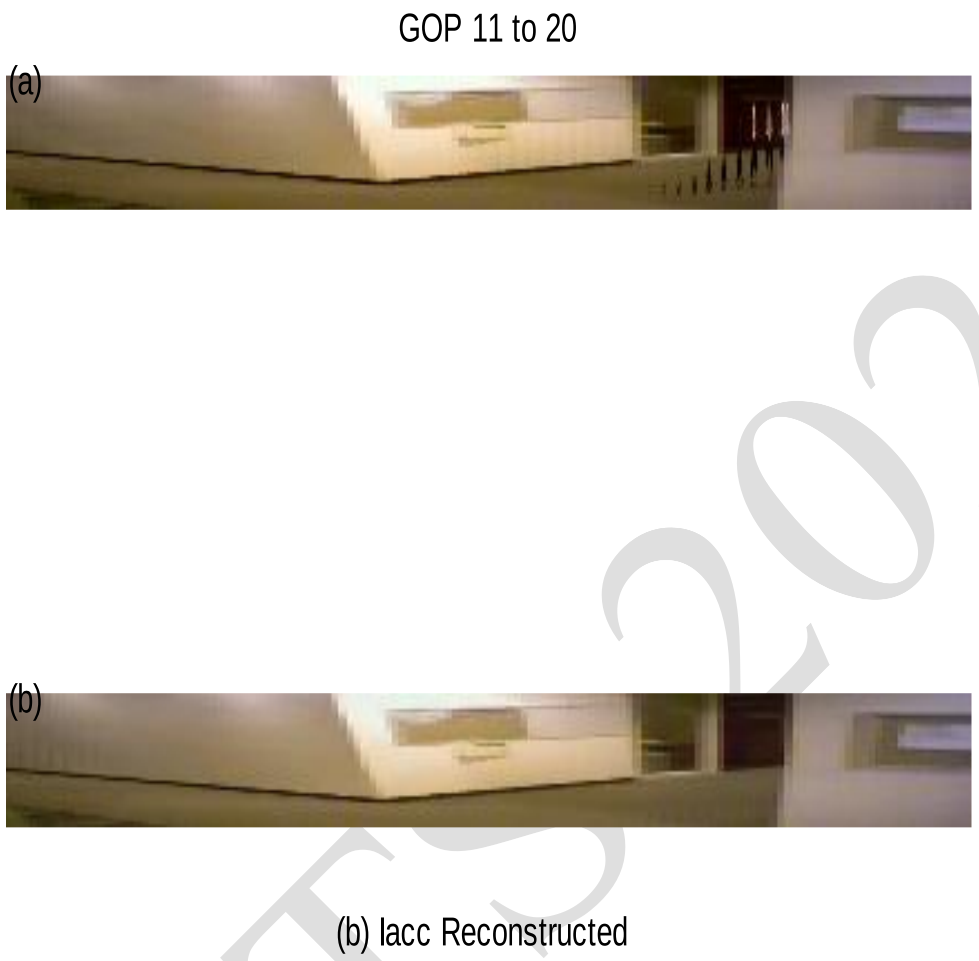

Fig. 5.9: Original Accordian imager and reconstructed image w.

PLOT
After decompression we can see figure 5.9 a and b.In both images we can see that there is very slight difference in color and contrast of both images.There is no large degradation in reconstructed video.

Step 4:Reconstruction of Original image from accordion image: In the figure 5.10 we have shown the image obtained after reconstruction .We can compare both video frames quality, color contrast and losses.

Original video before compression

Video obtained after decompression/decoding

Fig 5.10:

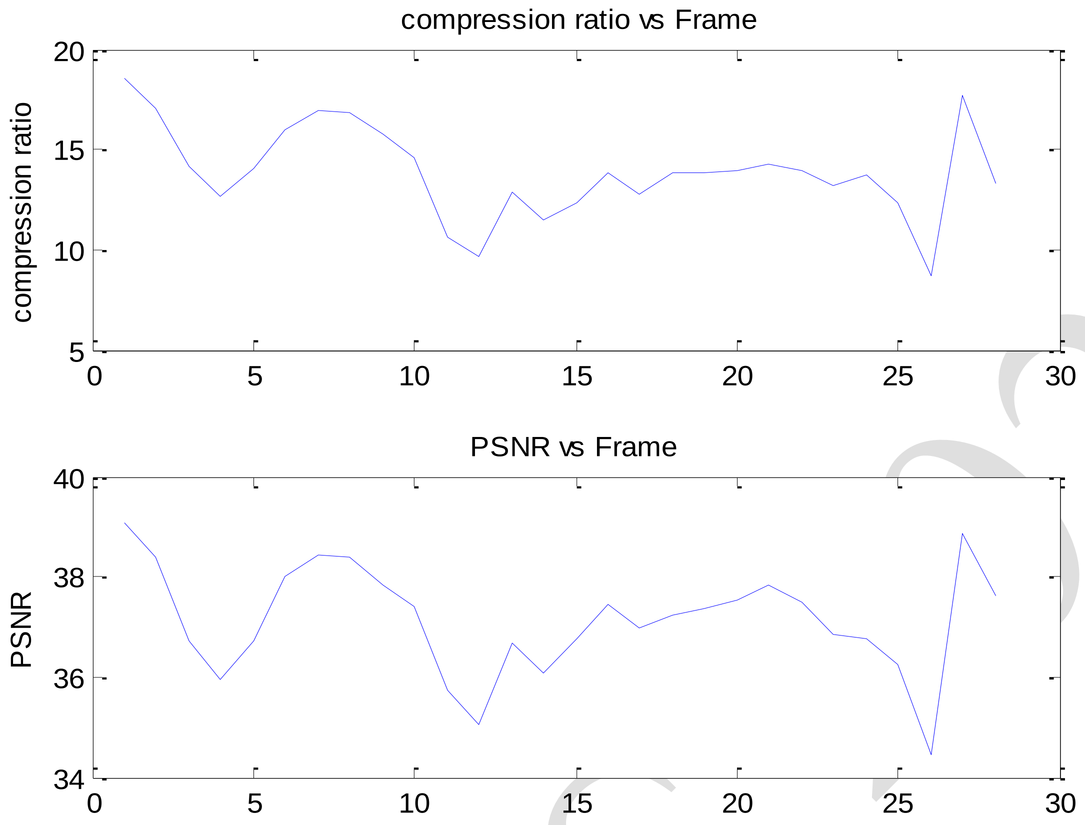

Fig 5.11:Plot for vipmen.avi video for compression ratio (above) and PSNR (below)

Figure 5.11 indicates the values of compression ration and PSNR obtained with respect to each frame.

Similar to ACC-DCT compression we performed only DCT compression on same videos to analyze our result in reference to compression ration and PSNR. WE obtained following results for DCT based video compression.

DCT video compression:(vipmen video)

Average PSNR= 35.8446

NET Compression=Original file/Compressed file = 9.6751

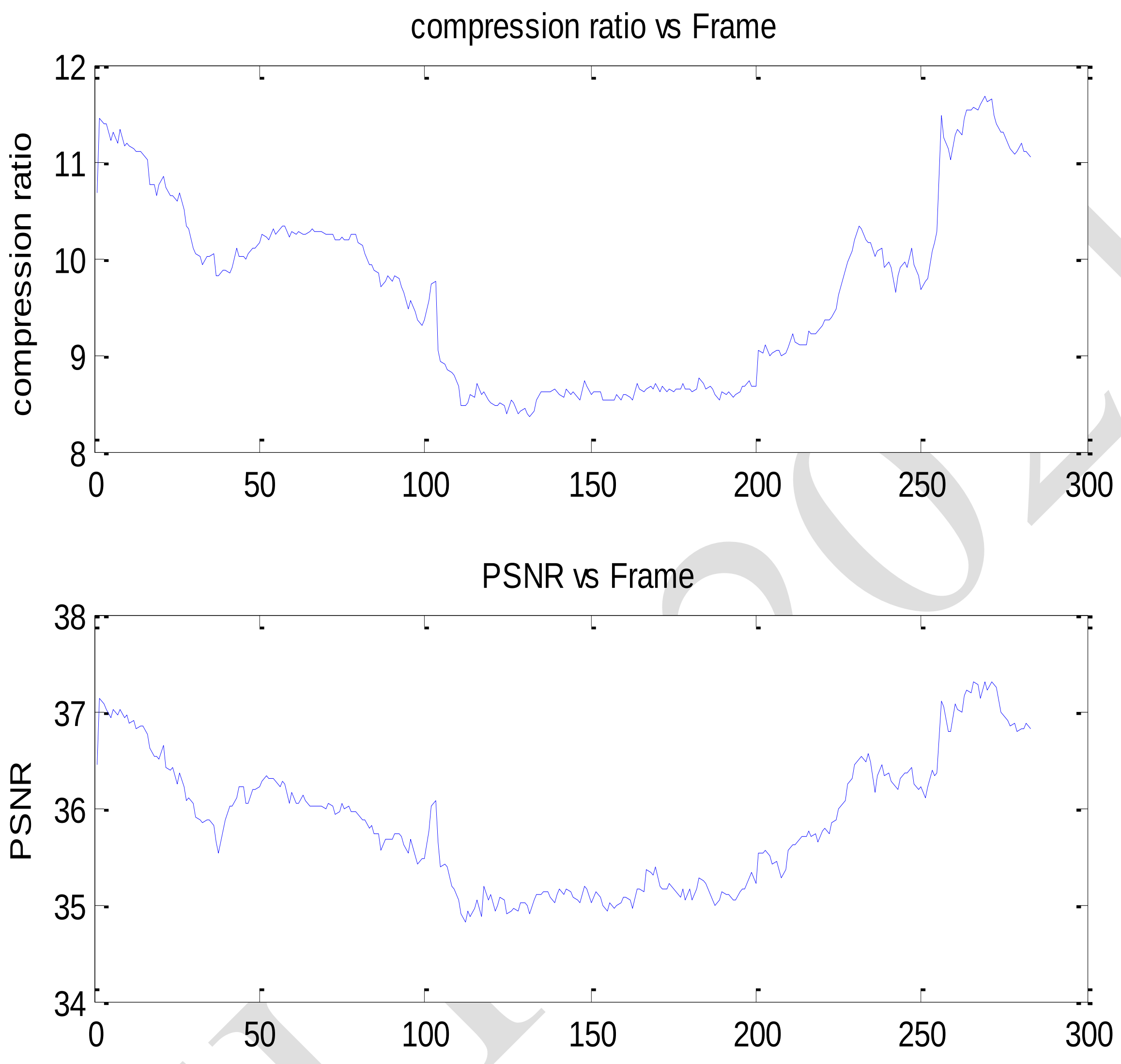

Fig. 5.12: Compression ration and PSNR using DCT video compression for vipmen.avi Accordian-DCT (viptraffic video)