# An Implementation of MPEG in Multi-terminal Video Compression

By

Saima Shaheen
(2006-NUST-MS PhD-CSE (E)-14)

Submitted to the Department of Computer Engineering
in fulfillment of the requirements for the degree of

Masters of Science
In
Computer Software Engineering

Thesis Supervisor

Dr. Muhammad Younas Javed

**College of Electrical & Mechanical Engineering**
**National University of Science & Technology**
**2010**

In the name of Almighty ALLAH, The Most
Gracious and Most Merciful

## Acknowledgements

All praise to the Almighty Allah, the most Merciful and the most Gracious, without Whose help and blessings, I would not have been able to complete this research. Many thanks to my project supervisor, Dr. Mohammad Younas Javed, whose constant motivation, unflagging efforts and uninvolved words of wisdom ever proved a lighthouse for me; it was earnestly felt whenever we swayed. I must also acknowledge the cooperation and perfect guidance of Dr. Muid Mufti that has encouraged me to complete this research work. I would like to acknowledge very specially the work of Mr. Zahid Ali (from Math works group). His MATLAB implementation for MPEG has helped me during the coding phase of this research work.

Acknowledgement is also due to my honorable and dedicated teachers for imparting enlightenment to me during the course of studies and afterwards for this project. I am also very thankful to my parents for their tacit and avowed support that has helped me through out my academic period.

*Dedicated to my;*
*Very co-operative family, especially to Afzaal Ahmed (my husband..)*

# Abstract

Video sensor networks (VSNe) are basically application related setups. These platforms have their own requirements and constraints. But, all these applications have one common problem of bandwidth requirement. Each node within a sensor network has to transmit its finding to a central receiver. Each terminal's information is important and carries a significant part of final results that must be transmitted to get best results. Bandwidth of the link is limited and it is a great problem to transmit all nodes findings within this available link capacity. Multimedia applications have huge textual as well as visual data. They, mostly comprise of repetitive patterns. Current video coding standards eliminate such data redundancy by exploiting spatial redundancy (within a video frame) as well as temporal redundancy (among video frames). These video codec standards are deployed on each sensor node that does not save the bandwidth requirement up to an optimum limit, since it does not exploit inter-sensor redundancy. Same video codec standards like MPEG can be deployed in a multi-terminal network by extending the transform coding of motion compensated coefficients among different sensor at a regular pattern. Deploying a standard video codec in this fashion exploits the inter-sensor redundancy, thus yields a great saving in required bandwidth for the resultant piece of information coming out of a wireless sensor network. This research work is an extension of MPEG, a video compression standard for a multi-camera setup.

Basically, in this research study camera sensor nodes of a wireless network are allowed to communicate in two strategies, strategy one allows minimum communication. While sensor nodes are allowed to communicate a bit more in strategy two. In both communication strategies, system has been designed to work in two working-modes: named as Scenario A and Scenario-B. Scenario A takes more processing time but gives large PSNR values while system in Scenario B can be deployed in a situation where quick system response is required provided degraded video quality can be compromised. Results in tabular as well as graphical forms have been formulated to evaluate the system performance in both working modes under both communication strategies. Overall system shows optimum performance at low bitrates.

# Table of Contents

# List of Figures

# List of Tables

# List of Abbreviations

| | |
|---|---|
| CP | Compression Percentage |
| CR | Compression Ratio |
| CSN | Camera Sensor Node |
| DCT | Discrete Cosine Transform |
| DSC | Distributed Source Coding |
| FDCT | Forward Discrete Cosine Transform |
| GOP | Group of Pictures |
| IDCT | Inverse Discrete Cosine Transform |
| MPEG | Moving Picture Expert Group |
| MTVC | Multi-Terminal Video Coding |
| MVSN | Multi-Video Sensor Network |
| PSNR | Peak Signal to Noise Ratio |
| VSN | Video Sensor Node |
| VSNe | Video Sensor Network |

# Report Chapters

# Chapter 1

# INTRODUCTION

In this chapter, the overview of the thesis is presented. This chapter holds, its importance providing the basis for research. It includes motivation, problem definition, objectives and goals of research.

## 1.1 Motivation

Multimedia applications constitute a major part of today's world of computing. These applications are designed to carry huge-sized visual and audio information that involves major issues concerning this data format transmission and storage. Since this information carries the repetitive patterns, there has always been a requirement for some standard to modify this data in some format carrying the information which is the most meaningful and important for the end-user.

A number of video compression techniques and standards have been developed to cater this problem; these comprise essential steps like applying various transforms and different coding techniques, each participating to enhance the resultant quality as well as removing the ambiguities and repetitions. Deploying this multimedia information on a wireless network is a great issue once again. Because the available link capacity is too short to carry this huge load. And then moving one step forward is consideration of multi-terminal scenarios, where we have to send the findings and measurements of each and every node of the network to a common link. Here this problem becomes more serious and very difficult to tackle.

## 1.2 Problem Definition

Since different sensors in a camera sensor network are capturing the same visual information from different angles so there is a great degree of correlation among these observations, known as inter-sequence statistical redundancy. Besides, this is the repetition of patterns in different frames of individual video sequence, called temporal correlation and then a high level of correlation also exists within a frame, called spatial redundancy.

Finally, for multi-camera sensor network, there should be some standard to cater all these three kinds of redundancies. Since each node in multi-camera scenario is equipped with standard video compression algorithm such as MPEG1, MPEG2, MPEG4, H261, H263, these all rely on efficient transform coding of motion-compensated frames. These standards eliminate temporal and spatial redundancies with in an individual video sequence. But these standards neglect the great degree of repetition that exists in resultant transmission stream due to inter-sequence correlation. There should be some mechanism to exploit this redundancy to save the final required bandwidth up to a greater extent.

## 1.3 Objectives and Goals

This research has mainly focused to exploit this correlation in a distributed fashion, keeping the communication among the sensors at minimum. A video compression standard, MPEG has been extended for a multi-camera environment in such a way to minimize the size of resultant bit-stream for transmission. Two communication strategies have been proposed for camera sensor nodes to communicate with one another. Moreover, system is capable of switching between two working modes. Since both the working modes reconstruct the output video sequences with different performance metrics. So each working mode can be deployed in a particular situation with different set of requirements. We proposed an architecture and methodology for improvement of compression ratio. As well as to keep the PSNR of reconstructed video sequence frames as high as possible.

Evaluation criteria for the results of proposed scheme will be:

a) Compression Ratio

b) Compression Percentage.

c) PSNR

d) Distortion (MSE)

Results for both communication strategies as well as for both working modes have been formulated. System performance has been evaluated and analyzed using MATLAB 7.0. And formulated results have been shown in tabular and graphical forms.

Chapter 2

# BACKGROUND

This chapter covers the background of this research work. The chapter starts with introduction to digital video, along with, its formats and video quality is discussed. Next section covers the compression and video compression importance and finally video compression standards are discussed. At the end working mechanism of multi camera setup is provided.

## 2.1 Digital Video

A digital video is actually a sequence of frames, which are normally presented at regular time intervals. A digital image is obtained by quantizing a continuous image both spatially and in amplitude. Digitization of the spatial coordinates is called image sampling, while digitization of the amplitude is called gray-level quantization. Suppose that a continuous image is denoted by g(x,y), where the amplitude or value of g at the point (x,y) is the intensity or brightness of an image at that point. The transformation of a continuous image to a digital image can then be expressed as:

$f(m,n,t) = Q[g(x_0 + m \text{ delta } x, y_0 + n \text{ delta } y)]$,

where Q is a quantization operator, $x_0$ and $y_0$ are the origin of image plane, m and n are the discrete values 0,1,2,….,and delta x and delta y are the sampling intervals in the horizontal and vertical directions, respectively. If this sampling process is extended to a third temporal direction, a sequence, f(m,n,t), is obtained.

$f(m,n,t) = Q[g(x_0 + m \text{ delta } x, y_0 + n \text{ delta } y, t_0 + t \text{ delta } t)]$,

Where t is the values 0, 1, 2, ….And delta t is the time interval. Each basic element of the image is called as a pixel or pel. Each individual image is called a frame. The frames are normally presented at a regular interval so that eye cans perceive fluid motion. For example, the NTSC (National Television System Committee) specified a temporal sampling rate of 30 frames/second and interlace 2 to 1.therefore, as a result of this spatio-temporal sampling, the digital signal exhibits high spatial and temporal correlation, just as the analog signals did before video data compression.

### 2.1.1 Digital video formats

In practical applications, most video signals are color signals. A color signal can be seen as a summation of light intensities of three primary wavelength bands. There are several color representations such as YCbCr, RGB and others. The YcbCr color representation is used for most video coding standards in compliance with the CCIR601 (International Radio Consultative Committee), common intermediate format (CIF), and SIF formats. The Y component specifies the luminance information and the Cb and Cr components specify the color information.

**Progressive and Interlaced**

Currently, most video signals that are generated by a TV camera are interlaced. These video signals are represented at 30 frames/second for an NTSC system. Each frame consists of two fields, the top field and bottom field, which are 1/60 of a second apart. In the display of an interlaced, the top field is scanned first and the bottom field is scanned next. The top and bottom fields are composed of alternating lines of the interlaced frame. Progressive video does not consist of fields, only frames. In an NTSC system, these frames are spaced 1/30 seconds apart. In contrast to interlaced video, every line within the frame is successively scanned.

**CCIR**

According to CCIR601 (CCIR is now known as ITU-R International Telecommunication Union-R), a color video source has three components: a luminance component (Y) and two color difference or chrominance components (Cb and Cr). The CCIR format has two options, I for the NTSC TV system and another for PAL TV system; both are interlaced. The NTSC format uses 525 lines/frame at 30 frames/second. The luminance frames of this format have 720 x 420 active pixels. The chrominance frames have two kinds of formats, one has 360 x 480 active pixels and is referred as the 4:2:2 format, while the other has 360 x 240 active pixels and is referred as the 4:2:0 format. The PAL format uses 625 lines/frame at 25 frames/second. Its luminance frame has 720 x 576 active pixels/frame and the chrominance frame has 360 x 576 active pixels /frame for the 4:2:2 format at 360x 288 pixels/frame for the 4:2:0 format, both at 25 frames/second.

**SIF (source input format)**

SIF has luminance resolution of 360x240 pixels/frame at 30 frames/second or 360x288 pixels/frame at 25 frames/second. For both cases, the resolution of the chrominance component is half of the luminance resolution in both horizontal and vertical dimensions. SIF can easily be obtained from a CCIR format using an appropriate anti-aliasing filter followed by sub sampling.

**CIF (common intermediate format)**

CIF is a non-interlaced format. Its luminance resolution has 352x288 pixels/frame at 30 frames/second and the chrominance has half the luminance resolution in both vertical and horizontal dimensions. Since its line value, 288,represents half the active lines in a PAL television signal, and its picture rate 30 frames/second, is the same as the NTSC television signal, it is a common intermediate format for both PAL and PAL-like systems and NTSC systems. In the NTSC systems, only a line number conversion is needed, while in the PAL and PAL-like systems only a picture rate conversion is needed. For low-bit-rate applications, the quarter-SIF (QSIF) or quarter-CIF (QCIF) formats may be used since these formats have only a quarter the number of pixels of SIF and CIF formats, respectively.

**ATSC (Advanced Television Standard Committee) DTV (digital television) format**

The concept of DTV consists of SDTV (standard-definition television) and HDTV (high definition television). Recently, in U.S., the FCC (federal communication commission) approved the ATSC-recommended DTV standard (ATSC, 1995). The DTV format is not included in the standard due to the divergent opinions of TV and computer manufactures. Rather, it has been agreed that the picture format will be decided by the future market. The ATSC-recommended DTV format including two kinds of formats:

SDTV and HDTV. The ATSC DTV standard includes the following 18 formats:

For HDTV: 1920x1080 pixels at 23.976/24 Hz, and 59.94/60 Hz progressive scan.

For SDTV: 704x480 pixels with 4:3 aspect ratio at 23.976/24 Hz, 29.97/30 Hz, 59.94/60 Hz progressive scan; 704x480 pixels with 16:9 aspect ratio at 23.976/24 Hz, 29.97/30 Hz, 59.94/60 Hz progressive scan; and 640x480 with 4:3 aspect ratio at 23.976/24 Hz, 29.97/30 Hz, 59.94/60 Hz progressive scan.

It is noted that all HDTV formats use square pixels and only part of SDTV format uses square pixels. The number of pixels/line vs. the number of line/frame is known as the aspect ratio.

## 2.2 Video Compression

Video is basically a three-dimensional array of color pixels. Two dimensions serve as spatial (horizontal and vertical) directions of the moving pictures, and one dimension represents the time domain. A data frame is a set of all pixels that correspond to a single time moment. Basically, a frame is the same as a picture. Video data contains spatial and temporal redundancy. Similarities can thus be encoded by merely registering differences within a frame (spatial), and/or between frames (temporal). Spatial encoding is performed by taking advantage of the fact that the human eye is unable to distinguish small differences in color as easily as it can perceive changes in brightness, so that very similar areas of color can be averaged out in a similar way to jpeg images. With temporal compression only the changes from one frame to the next are encoded as often a large number of the pixels will be the same on a series of frames. At the basic level, compression is performed when an input video stream is analyzed and information that is not useful to the viewer is discarded. Each event is then assigned a code - commonly occurring events are assigned few bits and rare events will have codes more bits. These steps are commonly called signal analysis, quantization and variable length encoding respectively.

Video compression is a tradeoff between disk space, video quality, and the cost of hardware required to decompress the video in a reasonable time. There are four methods for compression; discrete cosine transforms (DCT), vector quantization (VQ), fractal compression, and discrete wavelet transform (DWT).

**Discrete Cosine Transform (DCT)**

Discrete cosine transform, a lossy compression algorithm that samples an image at regular intervals, analyzes the frequency components present in the sample, and discards those frequencies which do not affect the image as the human eye perceives it. DCT is the basis of standards such as JPEG, MPEG, H.261, and H.263.

**Vector Quantization (VQ)**

Vector quantization is a lossy compression that takes an array of data, instead of individual values. It then generalizes all data, compressing redundant data, while at the same time retaining the desired object or data streams original intent.

**Fractal Compression (FC)**

Fractal compression is a form of VQ and is also a lossy compression. Compression is performed by locating self-similar sections of an image, then using a fractal algorithm to generate the sections.

**Discrete Wavelet Transform (DWT)**

Like DCT, discrete wavelet transform mathematically transforms an image into frequency components. The process is performed on the entire image, which differs from the other methods (DCT) that work on smaller pieces of the desired data. The result is a hierarchical representation of an image, where each layer represents a frequency band.

### 2.2.1 Why Compression?

Data/ image transmission and storage costs money. The more information being dealt with, the more it costs. In spite of this, most digital data are not stored in the most compact form. Rather, they are stored in any way which makes them easiest to use, such as: ASCII text form word processor, binary code that can be executed on a computer, individual samples from a data acquisition system, etc. typically, these easy-to-use encoding methods require data files about twice as large as actually needed to represent the information. Data compression is the general term for various algorithms and programs developed to address this problem. Likewise, an uncompressing program returns the information to its original form. Moreover, in today's world of computing, it is hardly possible to do without graphics, images and sound. Just by looking at the applications around us, the internet, development of video CDs (compact disks), video conferencing, and much more, all these applications use graphics and sound intensively. Uncompressed graphics, audio and video data consumes large amount of physical storage which for the case of uncompressed video, even present CD technology is unable to handle. Take for instance, if we want to display a TV-quality full motion video, how much of physical storage will be required? TV-quality video requires 720 kilobytes/frame (kbpf) displayed at 30 frames/second (fps). To obtain a full-motion effect, which means that one second of digitized video consumes approximately 22 MB (megabytes) of storage. A standard CD-ROM dick with 648 MB capacity and data transfer rate of 150 kbps could only provide a total of 30 seconds of video and would take 5 seconds to display a single frame. It is obviously just not acceptable. Transmission of uncompressed graphics, audio and video is a problem too. Expensive cables with high bandwidth are required to achieve satisfactory result, which is not feasible for the general market.

Thus, to provide feasible and cost effective solutions, most multimedia systems use compression techniques to handle graphics, audio and video data streams.

### 2.2.2 Video Compression Standards

MPEG is an ISO/IEC working group, established in 1988 to develop standards for digital audio and video formats. There are five MPEG standards being used or in development. Each compression standard was designed with a specific application and bit rate in mind, although MPEG compression scales well with increased bit rates. They include:

**MPEG-1**

It is designed for up to 1.5 Mbit/sec, a standard for the compression of moving pictures and audio. This was based on CD-ROM video applications, and is a popular standard for video on the Internet, transmitted as .mpg files. In addition, level 3 of MPEG-1 is the most popular standard for digital compression of audio--known as MP3. MPEG-1 is the standard of compression for Video CD, it is the most popular video distribution format.

**MPEG-2**

It has been designed for range between 1.5 and 15 Mbit/sec, a standard on which Digital Television set top boxes and DVD compression is based. It is based on MPEG-1, but designed for the compression and transmission of digital broadcast television. The most significant enhancement from MPEG-1 is its ability to efficiently compress interlaced video. MPEG-2 scales well to HDTV resolution and bit rates, obviating the need for an MPEG-3.

**MPEG-4**

This is a standard for multimedia and Web compression. MPEG-4 is based on object-based compression, similar in nature to the Virtual Reality Modeling Language. Individual objects within a scene are tracked separately and compressed together to create an MPEG4 file. This results in very efficient compression that is very scalable; from low bit rates to very high. It also allows developers to control objects independently in a scene, and therefore introduce interactivity. DivX is a software application that uses the MPEG-4 standard to compress digital video, so it can be downloaded over a DSL/cable modem connection in a relatively short time with no reduced visual quality. The latest version of the codec, DivX 4.0, is being developed jointly by DivX Networks and the open source community. DivX works on Windows 98, ME, 2000, CE, Mac and Linux.

**MPEG-7**

This standard, currently under development, is also called the Multimedia Content Description Interface. When released, the group hopes the standard will provide a framework for multimedia content that will include information on content manipulation, filtering and personalization, as well as the integrity and security of the content. Contrary to the previous MPEG standards, which described actual content, MPEG-7 will represent information about the content.

**MPEG-21**

The work on this standard, also called the Multimedia Framework, has just begun. MPEG-21 will attempt to describe the elements needed to build an infrastructure for the delivery and consumption of multimedia content, and how they will relate to each other.

DV is a high-resolution digital video format used with video cameras and camcorders. The standard uses DCT to compress the pixel data and is a form of lossy compression. The resulting video stream is transferred from the recording device via FireWire (IEEE 1394), a high-speed serial bus capable of transferring data up to 50 MB/sec.

**H.261**

It is an ITU standard designed for two-way communication over ISDN lines (video conferencing) and supports data rates which are multiples of 64Kbit/s. The algorithm is based on DCT and can be implemented in hardware or software and uses intraframe and interframe compression. H.261 supports CIF and QCIF resolutions.

**H.263, H.263 Version 2 (H.263+), H.263++ and H.261**

The H.263 video coding standard is specifically designed for very low bit rate applications such as video conferencing. Its technical content was completed in late 1995 and the standard was approved in early 1996. It is based on the H.261 standard with several added features: unrestricted motion vectors, syntax based-arithmetic coding, advanced prediction and P-B frames. The H.263 version 2 video coding standards, also known as "H.263+", was approved in January 1998 by the ITU-T. H.263+ includes a number of optional features based on the H.263. These new optional features are added to provide improved coding efficiency, a flexible video format, scalability and backward compatible supplemental enhancement information. H.263++ is the extension of H.263+ and is currently scheduled to be completed in the year 2000. H.26L is along term project which is looking for more efficient video coding algorithms.

## 2.3 Multi-Terminal Video Coding (MTVC)

A wireless sensor network consists of many tiny, low-power and cheap wireless sensors. Unlike personal computers or the Internet, which are designed to support all types of applications, sensor networks are usually mission-driven and application specific e.g. designed for the detection of biological agents and toxic chemicals; environmental measurement of temperature, pressure and vibration; or real-time area video surveillance. Sensor nodes in this network must operate under a set of unique constraints and requirements.

Unlike many other wireless devices (e.g., cellular phones, PDAs, and laptops), in which energy can be recharged from time to time, the energy provisioned for a wireless sensor node is not expected to be renewed throughout its mission. The limited amount of energy available to wireless sensors has a significant impact on all aspects of a wireless sensor network, from the amount of information that the node can process, to the volume of wireless communication it can carry across large distances. Realizing the great promise of sensor networks requires more than a mere advance in individual technologies; it relies on many components working together in an efficient, unattended, comprehensible, and trustworthy manner.

### 2.3.1 Distributed Source Coding (DSC)

One of the enabling technologies for sensor networks is Distributed Source Coding (DSC), which refers to the compression of multiple correlated sensors outputs independently that do not communicate with each other, these sensors send their compressed outputs to a central point (e.g., the base station) for joint decoding.

For DSC, consider a wireless video sensor network consisting of clusters of low cost Video Sensor Nodes (VSNs), an Aggregation Node (AN) for each cluster and a base station for surveillance applications. The lower tier VSNs are used for data acquisition and processing; the upper tier ANs are used for data fusion and transmitting information out of the network. This type of network is expected to operate unattended over an extended period of time.VSN and AN in this setup power consumption cause severe system constraints; in addition to this is traditional video processing applied to sophisticated video encoders will not be suitable for use on a VSN. This is because, under the traditional broadcast paradigm the video encoder is the computational workhorse of the video codec; consequently, computational complexity is dominated by the motion estimation operation. The decoder, on the other hand, is a relatively lightweight device operating in a “slave” mode to the encoder. The severe power constraints at VSNs thus bring about the following basic requirements:

a) An extremely low-power and low-complexity wireless video encoder, which is critical to prolonging the lifetime of a wireless video sensor node.

b) A high ratio of compression efficiency, since bit rate directly impacts transmission power consumption at a node.

DSC allows a many-to-one video coding paradigm that effectively swaps encoder-decoder complexity with respect to conventional (one-to-many) video coding, thereby representing a fundamental conceptual shift in video processing. Under this paradigm, the encoder at each VSN is designed as simply and efficiently as possible, while the decoder at the base station is powerful enough to perform joint decoding.

Furthermore, each VSN can operate independently of its neighbors; consequently, a receiver is not needed for video processing at a VSN, which enables the system to save a substantial amount of hardware cost and communication (i.e., receiver) power. But practically depending also on the nature of the sensor network, the VSN might still need a receiver to take care of other operations, such as routing, control, and synchronization, but such a receiver will be significantly less sophisticated.

Under this new DSC paradigm, a challenging problem is to achieve the same efficiency (e.g., joint entropy of correlated sources) as traditional video coding, while not requiring sensors to communicate with each other. This seems possible because correlation exists among readings from closely-placed neighboring sensors and the decoder can exploit such correlation with DSC – this is done at the encoder with traditional video coding. As an example, suppose we have two correlated 8-bit grayscale images X and Y. The correlation of x and y is characterized as x assumes only eight different values around y. So joint coding of x would take three bits. But in DSC, we simply take modulo of pixel value x with respect to eight, which also reduces the required bits to three. For example, if x = 121 and y = 119. Instead of transmitting both x and y at 8 b/p without loss, we transmit y = 119 and x0 = x (mod 8) = 1 in distributed coding. Consequently, x0 indexes the set that x belongs to, i.e.

x 2 f 1; 8 + 1; 16 + 1; : : : ; 248 + 1 g ,

And the joint decoder picks the element x = 120 + 1 closest to y = 119.

DSC is only one of the communication layers in a network, and its interaction with the lower communication layers, such as the transport, the network, the medium access control (MAC), and the physical layers, which is the focus of this special issue, is crucial for exploiting the promised gains of DSC. DSC cannot be used without proper synchronization between the nodes of a sensor network, i.e., several assumptions are made for the routing and scheduling algorithms and their connection to the utilized DSC scheme.

### 2.3.2 Applications of DSC in Sensor Networks

Two information theories Wyner-Ziv theorem and Slepian-wolf theorem proposed in 1970s, provided theoretical basis of DSC in wireless sensor networks. Slepian-Wolf theorem addresses the lossless case of DSC while Wyner-Ziv coding is for lossy DSC case.

**(a) Wyner-Ziv Theorem**

In sensor network applications, dealing with continuous sources; problem of rate distortion with side information at the decoder arises. The question to ask is how many bits are needed to encode X under the constraint that the average distortion between X and the coded version $\hat{X}$ is E f d(X; $\hat{X}$ ) g · D, assuming the side information Y is available at the decoder but not at the encoder. This problem, first considered by Wyner and Ziv, is one instance of DSC with Y available uuencoded as side information at the decoder. It generalizes the setup that coding of discrete X is with respect to a fidelity criterion rather than lossless. For both discrete and continuous alphabet cases and general distortion metrics, Wyner and Ziv gave the rate-distortion function R¤ WZ (D) for this problem. Wyner-Ziv coding generalizes the setup of Slepian-Wolf coding in that coding of X is with respect to a fidelity criterion rather than lossless.

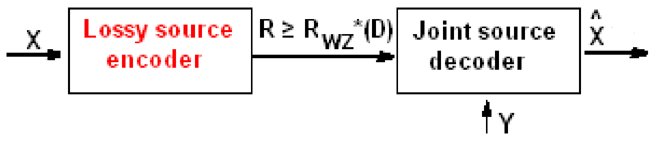
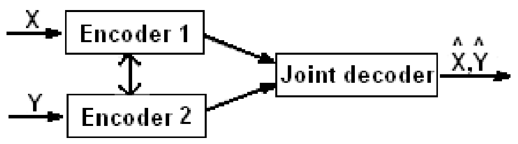


**Figure 2.1:** Wyner-Ziv coding

### (b) Slepian-Wolf Theorem



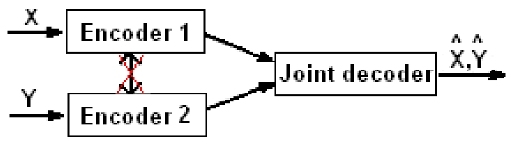


**Figure 2.2 (a):** Joint encoding of X and Y

**Figure 2.2 (b):** Distributed encoding of X and Y

Slepian wolf coding is referred to as lossless distributed source coding since it considers that the two statistically dependent sequences are perfectly reconstructed at a joint decoder (neglecting the arbitrarily small probability of decoding error),thus approaching the lossless case. They theoretically showed that separate encoding at each independent encoder with increased complexity at the joint decoder is as efficient as joint encoding for lossless compression. The Slepian-Wolf theorem addresses the case where two statistically dependent discrete random sequences, independently and identically distributed, X and Y, are independently encoded, and thus not jointly encoded as in the largely deployed predictive coding solution. The Slepian–Wolf theorem states that the minimum rate to encode the two (correlated) sources is the same as the minimum rate for joint encoding, with an arbitrarily small error probability.

Let f (Xi; Yi) g 1 i=1 be a sequence of independent and identically distributed drawings of a pair of correlated discrete random variables X and Y. For lossless compression with $\hat{X} = X$ and $\hat{Y} = Y$ after decompression, from Shannon's source coding theory that a rate given by the joint entropy H(X; Y) of X and Y is sufficient if we are encoding them together from Figure 2.2(a). For example, Y can be compressed into H(Y ) bits per sample, and based on the complete knowledge of Y at both the encoder and the decoder, X can be compressed into H(X j Y ) bits per sample. One simple way is to do separate coding of X and Y with rate R = H(X) + H(Y), which is greater than H(X; Y) when X and Y are correlated. In their research paper, Slepian and Wolf showed that R = H(X; Y) is sufficient even for separate encoding of correlated sources as from Figure 2.2(b). Specifically, the Slepian-Wolf theorem says that the achievable region of DSC for discrete sources X and Y is given by R1 ¸ H(X j Y ); R2 ¸ H(Y j X) and R1 + R2 ¸ H(X; Y ), as shown in Figure 2.3. The proof of the Slepian-Wolf theorem is based on random binning. Binning is a key concept in DSC and refers to as partitioning the space of all possible outcomes of a random source into disjoint subsets or bins.

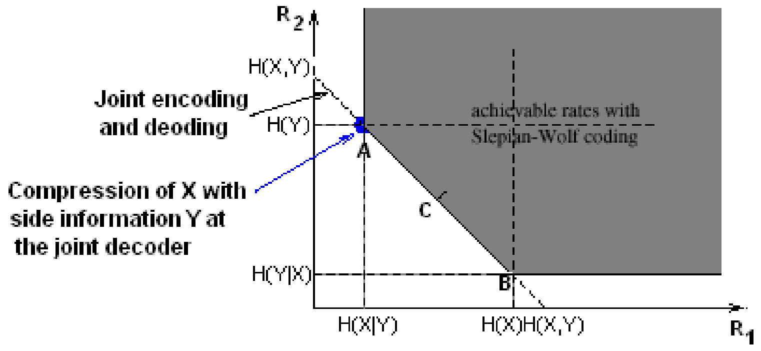

$R_2$
H(X,Y)
Joint encoding and deoding
H(Y)
achievable rates with Slepian-Wolf coding
A
Compression of X with side information Y at the joint decoder
C
H(Y|X)
B
H(X|Y)
H(X)H(X,Y)
$R_1$

**Figure 2.3:** The Slepian-Wolf rate region

## 2.4 Summary

In this chapter, the background knowledge of the research has been presented. The basis of digital video has been described, along with various video compression standards and the concept of multi-terminal video coding. The next chapter will provide detail of MPEG and its working steps.

Chapter 3

# Moving Picture Expert Group (MPEG)

This chapter covers Moving Picture Expert Group (MPEG), a video compression standard in details. It includes basic feature introduction and technical details of codec (encoder/decoder) architecture.

## 3.1 Introduction

MPEG stands for the Moving Picture Experts Group. MPEG is an ISO/IEC working group, established in 1988 to develop standards for digital audio and video formats. There are five MPEG standards being used or developed. Each compression standard was designed with a specific application and bit rate.

MPEG is the standard for coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbps. To support a wide range of application profiles the user can specify a set of input parameters including flexible picture size and frame rate. MPEG-1 was developed for multimedia CD-ROM applications. Important features provided by MPEG-1 include frame based random access of video, fast forward/fast reverse search through compressed bit streams reserve playback of video and edit ability of compressed bit streams.

The algorithm employed by MPEG-1 doesn’t provide a lossless coding scheme. However the standard can support a variety of input formats and be applied to a wide range of applications. The main purpose of MPEG-1 video is to code moving image sequences or video signals. To achieve a high compression ratio, both intraframe redundancy and interframe redundancy should be exploited. To satisfy the requirement of random access, intraframe coding is also implemented time to time (for I frames). Therefore the MPEG-1 video algorithm is mainly based on discrete cosine transform (DCT) coding and interframe motion compensation. The DCT coding is used to remove the intraframe redundancy and motion compensation is used to remove the interframe redundancy. With regard to input picture format, MPEG-1 allows progressive pictures only but offers great flexibility in size up to 4095 x 4095 pixels.

## 3.2 History

The convener of the MPEG group is Leonardo Chiariglione, aka the father of MPEG, who founded the group in January 1988 with the first meeting consisting of about 15 experts on compression technology. Modeled on the successful collaborative approach and the compression technologies developed by the Joint Photographic Experts Group and CCITT's Experts Group on Telephony (creators of the JPEG image compression standard and the H.261 standard for video conferencing respectively) the Moving Picture Experts Group (MPEG) working group was established in January 1988. MPEG was formed to address the need for standard video and audio formats, and build on H.261 to get better quality through the use of more complex encoding methods. Development of the MPEG-1 standard began in May 1988. 14 video and 14 audio codec proposals were submitted by individual companies and institutions for evaluation. The codecs were extensively tested for computational complexity and subjective (human perceived) quality, at data rates of 1.5 Mbit/s. This specific bit rate was chosen for transmission over T-1/E-1 lines and as the approximate data rate of audio CDs. The codecs that excelled in this testing were utilized as the basis for the standard and refined further, with additional features and other improvements being incorporated in the process. After 20 meetings of the full group in various cities around the world, and 4½ years of development and testing, the final standard (for parts 1-3) was approved in early November 1992 and published a few months later. A largely complete draft standard was produced in September 1990, and from that point on, only minor changes was introduced. The standard was finished with the 6 November 1992 meeting. The Berkeley Plateau Multimedia Research Group developed a MPEG-1 decoder in November 1992. In July 1990, before the first draft of the MPEG-1 standard had even been written, work began on a second standard, MPEG-2, intended to extend MPEG-1 technology to provide full broadcast-quality video (as per CCIR 601) at high bitrates (3 - 15 Mbit/s), and support for interlaced video. Due in part to the similarity between the two codecs, the MPEG-2 standard includes full backwards compatibility with MPEG-1 video, so any MPEG-2 decoder can play MPEG-1 videos.

## 3.3 Features

### 3.3.1 Resolution/Bitrate

MPEG-1 supports resolutions up to 4095×4095 (12-bits), and bitrates up to 100 Mbit/s. MPEG-1 videos are most commonly seen using Source Input Format (SIF) resolution: 352x240, 352x288, or 320x240. These low resolutions, combined with a bit rate less than 1.5 Mbit/s, make up a Constrained Parameters Bit stream (CPB), later renamed the

"Low Level" (LL) profile in MPEG-2. This is the minimum video specifications any decoder should be able to handle, to be considered MPEG-1 compliant. This was selected to provide a good balance between quality and performance, allowing the use of reasonably inexpensive hardware of the time.

### 3.3.2 Layered Structure Based on Group of Pictures

The MPEG coding algorithm is a full-motion-compensated DCT and DPCM hybrid coding algorithm. In MPEG coding, the video sequence is first divided into groups of pictures (GOP). Each GOP may include three types of pictures: Intracoded (I) pictures, Predictive-coded (P) and Bidirectional predictive-coded (B) pictures. These picture types serve different purposes. The most important are I-frames.

**I-frames**

I-frame is an abbreviation for Intra-frame, since they can be decoded independently of any other frames. They are also known as I-pictures, or key frames. I-frames can be considered effectively identical to baseline JPEG images. High-speed seeking through an MPEG-1 video is only possible to the nearest I-frame. When cutting a video it is not possible to start playback of a segment of video before the first I-frame in the segment.

**P-frames**

P-frames may also be called forward-predicted frames, or inter-frames. P-frames exist to improve compression by exploiting the temporal (over time) redundancy in a video. P-frames store only the difference in image from the frame (either an I-frame or P-frame) immediately preceding it (this reference frame is also called the anchor frame). The difference between a P-frame and its anchor frame is calculated using motion vectors on each macro block of the frame. Such motion vector data will be embedded in the P-frame for use by the decoder. A P-frame can contain any number of intra-coded blocks, in addition to any forward-predicted blocks.

**B-frames**

B-frames may also be known as backwards-predicted frames. B-frames are quite similar to P-frames; except they can make predictions using both the previous and future frames (i.e. two anchor frames).

It is therefore necessary for the player to first decode the next I- or P- anchor frame sequentially after the B-frame, before the B-frame can be decoded and displayed. This makes B-frames very computationally complex, requires larger data buffers, and causes an increased delay on decoding and during encoding. No other frames are predicted from a B-frame. Because of this, a very low bit rate B-frame can be inserted, where needed, to help control the bit rate. If this was done with a P-frame, future P-frames would be predicted from it and would lower the quality of the entire sequence. However, similarly, the future P-frame must still encode all the changes between it and the previous I- or P- anchor frame (a second time) in addition to much of the changes being coded in the B-frame. B-frames can also be beneficial in videos where the background behind an object is being revealed over several frames, or in fading transitions, such as scene changes. A B-frame can contain any number of intra-coded blocks and forward-predicted blocks, in addition to backwards-predicted, or bidirectional predicted blocks.

The distance between two nearest I-frames is denoted by N, which is the size of GOP. The distance between two nearest anchor frames is denoted by M. MPEG-1 most commonly uses a GOP size of 15-18. i.e. 1 I-frame for every 14-17 non-I-frames. With more intelligent encoders, GOP size is dynamically chosen, up to some pre-selected maximum limit. Limits are placed on the maximum number of frames between I-frames due to decoding complexity, decoder buffer size, recovery time after data errors, and seeking ability, but large values of M and N may cause error propagation. The structure of MPEG implies that if an error occurs within I-frame data, it will be propagated through all frames in the GOP. Similarly, an error in a P-frame will affect the related P and B-frames, while B-frames error will be isolated.

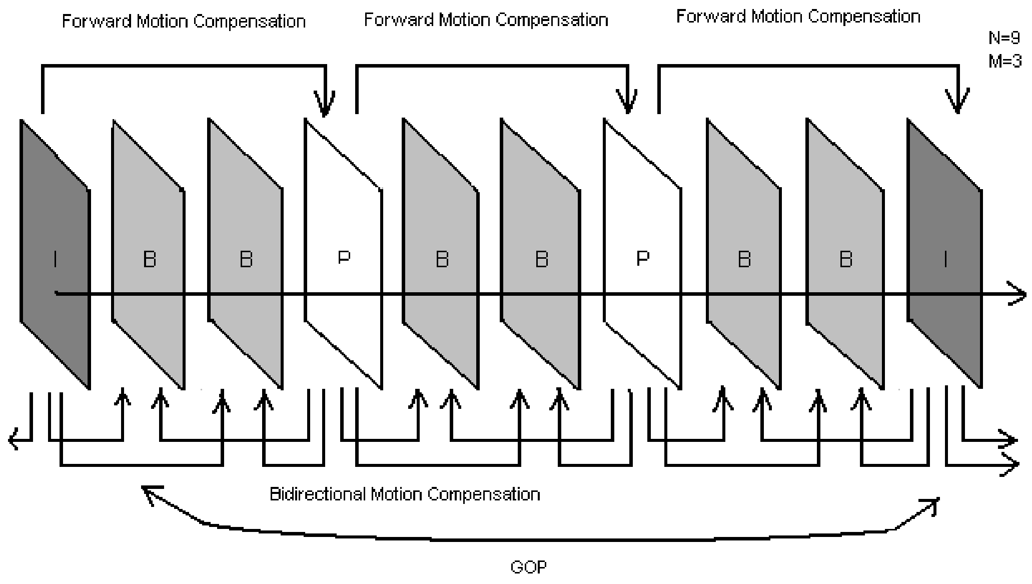

Forward Motion Compensation
Forward Motion Compensation
Forward Motion Compensation
N=9
M=3
I
B
B
P
B
B
P
B
B
I
Bidirectional Motion Compensation
GOP

**Figure 3.1:** A GOP of video sequence.

### 3.3.3 Intraframe and Interframe Coding

One of the most powerful techniques for compressing video is interframe compression. Interframe compression uses one or more earlier or later frames in a sequence to compress the current frame, while intraframe compression uses only the current frame, which is the approach used in image compression.

The interframe coding method works by comparing each frame in the video with the previous one. If the frame contains areas where nothing has moved, the encoder simply issues a short command that copies that part of the previous frame, bit-for-bit, into the next one. If sections of the frame move in a simple manner, the encoder emits a (slightly longer) command that tells the decoder to shift, rotate, lighten, or darken the copy. Interframe compression works well for programs that will simply be played back by the viewer, but can cause problems if the video sequence needs to be edited. The MPEG video uses the macroblock structure for motion compensation; i.e., for each 16 x 16 macroblock only one or sometimes two motion vectors are transmitted. The motion vectors for any block are found within a search window that can be up to 512 pixels in each direction. Also the matching can be done at half-pixel accuracy, where the half-pixel values are computed by averaging the full-pixel values. For interframe coding, the prediction differences or error images are coded and transmitted with motion information.

Since interframe compression copies data from one frame to another, if the original frame is simply cut out or lost in transmission, the following frames cannot be reconstructed properly.

### 3.3.4 Motion Estimation and Compensation

In interframe coding, motion estimation and compensation have become powerful techniques to eliminate the temporal redundancy due to high correlation between consecutive frames.

Most of the motion estimation algorithms make the following assumptions:

1. Objects move in translation in a plane that is parallel to the camera plane, i.e., the effects of camera zoom, and object rotations are not considered.
2. Illumination is spatially and temporally uniform.
3. Occlusion of one object by another, and uncovered background are neglected.

There are two mainstream techniques of motion estimation:

a) **Pel-Recursive Algorithm (PRA)**

**b) Block-Matching Algorithm (BMA)**

PRAs are iterative refining of motion estimation for individual pels by gradient methods. BMAs assume that all the pels within a block have the same motion activity. BMAs estimate motion on the basis of macro blocks (rectangular block) and produce one motion vector for each block. PRAs involve more computational complexity and less regularity, so they are difficult to realize in hardware. In general, BMAs are more suitable for a simple hardware realization because of their regularity and simplicity.

Figure 3.2 illustrates a process of block-matching algorithm. In a typical BMA, each frame is divided into blocks, each of which consists of luminance and chrominance blocks. Usually, for coding efficiency, motion estimation is performed only on the luminance block. Each luminance block in the present frame is matched against candidate blocks in a search area on the reference frame. These candidate blocks are just the displaced versions of original block. The best (lowest distortion, i.e., most matched) candidate block is found and its displacement (motion vector) is recorded. In a typical interframe coder, the input frame is subtracted from the prediction of the reference frame. Consequently the motion vector and the resulting error can be transmitted instead of the original luminance block; thus interframe redundancy is removed and data compression is achieved. At receiver end, the decoder builds the frame difference signal from the received data and adds it to the reconstructed reference frames. The summation gives an exact replica of the current frame. The better the prediction the smaller the error signal and hence the transmission bit rate.

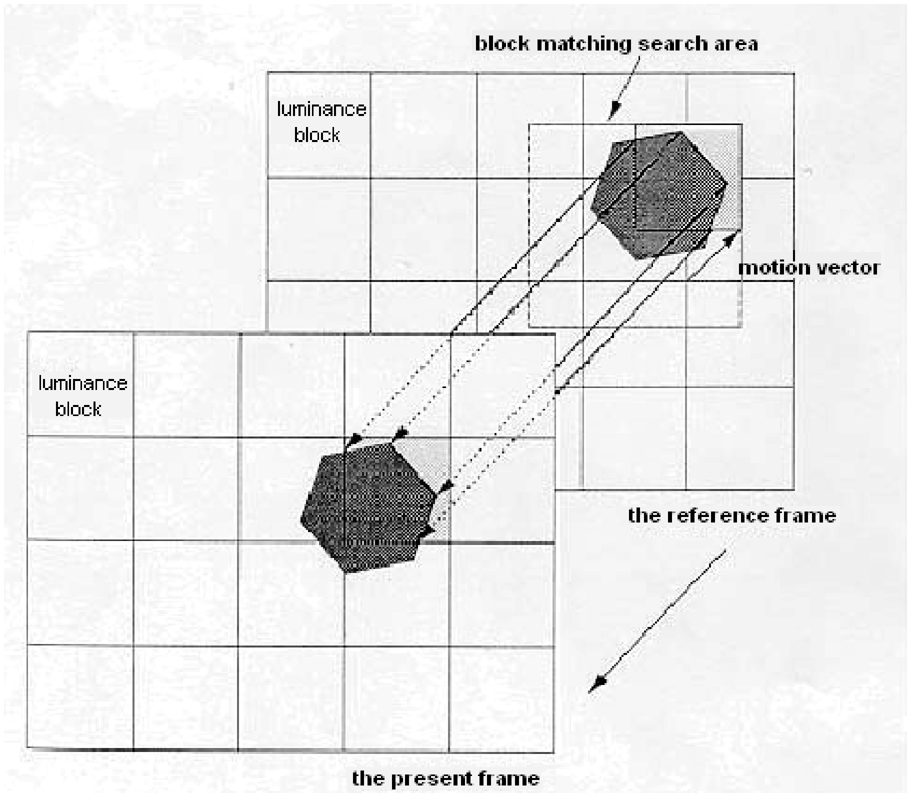


**Figure 3.2:** Block matching

### 3.3.5 Block Matching Algorithms (BMA)

Interframe predictive coding is used to eliminate the large amount of temporal and spatial redundancy that exists in video sequences and helps in compressing them. The use of the knowledge of the displacement of an object in successive frames is called Motion Compensation. There are a large number of motion compensation algorithms for interframe predictive coding, a class of which is called the Block Matching Algorithms. These algorithms estimate the amount of motion on a block by block basis, i.e. for each block in the current frame, a block from the previous frame is found, that is said to match this block based on a certain criterion.

**Search Criteria**

Different kinds of algorithms use different criteria for comparison of blocks. There are a number of criteria to evaluate the level of a match and some of them are:

a) Cross Correlation Function (CCF)

b) Pel Difference Classification (PDC)

c) Mean Absolute Difference (MAD)

d) Mean Squared Difference (MSD)

e) Integral Projection (IP)

**Search Strategy**

The searching strategy is another important issue to deal with in block matching; there are several search strategies as:

a) Full Search (FS)

b) Three Step Search (TSS)

c) Two Dimensional Logarithmic Search (TDL)

d) Four Step Search (FSS)

e) Orthogonal Search Algorithm (OSA)

f) One at a Time Algorithm (OTA)

g) Cross Search Algorithm (CSA)

h) Spiral Search (SS)

i) Hierarchical Search Block Matching Algorithms (HSBMA)

j) Binary Search (BS)

**Binary Search Algorithm**

This is also one of the algorithms that are very popular for motion estimation and in fact it is used for motion estimation by MPEG-Tool. The basic idea behind this algorithm is to divide the search window into a number of regions and do a full search only in one of these regions. It may be described as:

Step 1: The MAD is evaluated on a grid of 9 pixels that include the centre, the four corners of the search window and four pels at the boundaries. The search window is divided into regions based on these points.

Step 2: A full search is performed in the region corresponding to the point with the smallest MAD.

The convergence of the algorithm may be viewed in Figure 3.3. The pels that lie between the dashed lines are never considered. Hence, although the Binary search requires fewer comparisons (the worst case scenario for this search window is 33 comparisons), its performance is not very good because of this zone of pixels that are never considered.

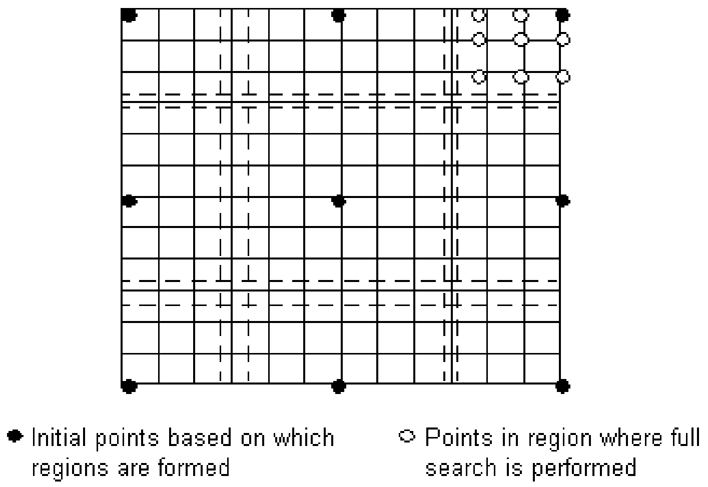

● Initial points based on which regions are formed
○ Points in region where full search is performed

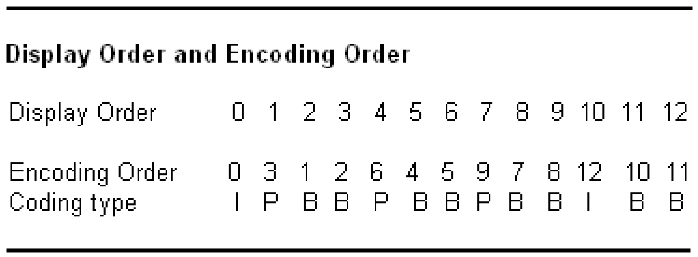
**Figure 3.3:** Example path for convergence of Binary Search

### 3.3.6 Codec Architecture

#### Encoding Process:

The typical MPEG-1 video encoder structure is shown in Figure 3.4.Since the encoding order is different from the display order, the input sequence has to be reordered for encoding for example, if we choose the GOP size (N) to be 12, and the distance between two nearest anchor fames (M) to be 3, the display order and encoding order are as shown in Table 3.1.

**Table 3.1: Display order and encoding order of a GOP**



**Display Order and Encoding Order**

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Display Order | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| Encoding Order | 0 | 3 | 1 | 2 | 6 | 4 | 5 | 9 | 7 | 8 | 12 | 10 | 11 |
| Coding type | I | P | B | B | P | B | B | P | B | B | I | B | B |

It should be noted that in the encoding order or in the bit stream the first frame in a GOP is always an I-picture. In the display order the first frame can be either in I-picture or the first B-picture of he consecutive series of B-picture which immediately precedes the first I-picture, and the last picture in the GOP is an anchor picture, either an I- or P-picture. The first GOP always starts with an I-picture and, as a consequence, this GOP will have fewer B- pictures than the other GOPs.

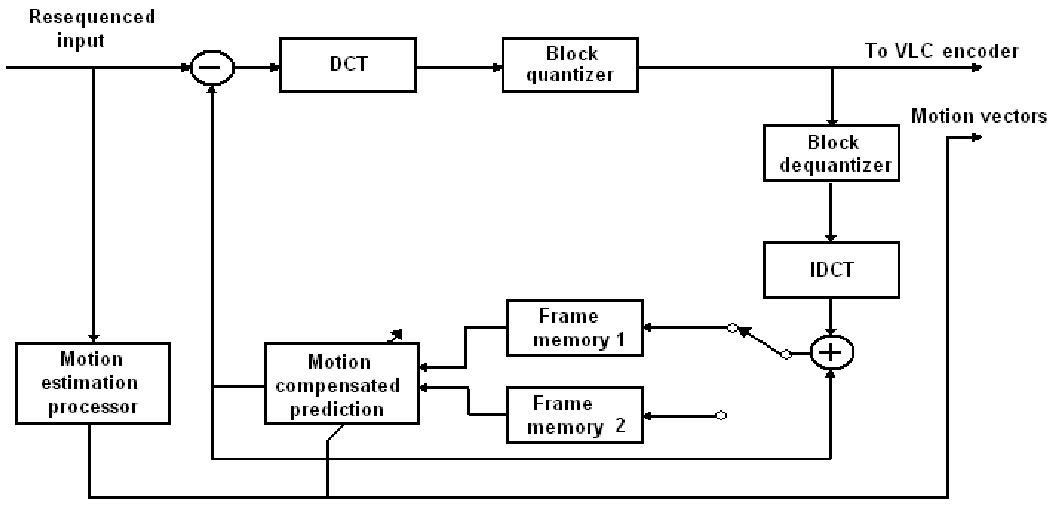
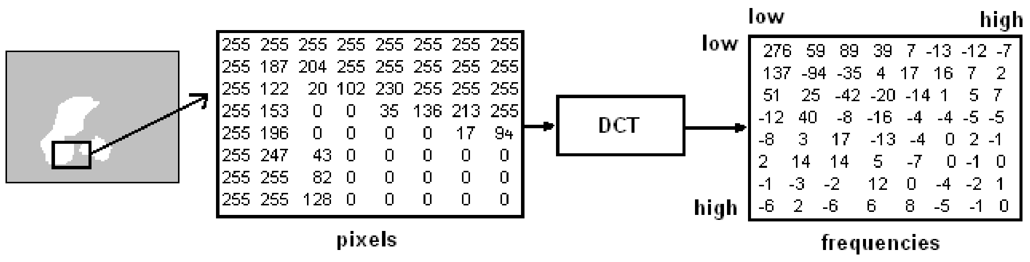


**Figure 3.4:** MPEG Encoder Structure

### DCT Transform

The image to be coded is first partitioned into 8x8 blocks. Each 8x8 block is encoded by applying a Forward Discrete Cosine Transform (FDCT), The FDCT process (by itself) is theoretically lossless, and can be reversed by applying an Inverse DCT (IDCT) to reproduce the original values.

The FDCT process converts this 8x8 block of uncompressed pixel values into an 8x8 indexed array of frequency coefficient values. This is frequency domain representation of the block as shown in Figure 3.5. The goal of the transformation is to decorrelate the block data so that the resulting transform coefficients can be coded more efficiently. One of these coefficients is the (statistically high in variance) **DC coefficient**, which represents the average value of the entire 8x8 block. The other 63 coefficients are the statistically smaller **AC coefficients**, which are positive or negative values each representing sinusoidal deviations from the flat block value represented by the DC coefficient. Since the DC coefficient value is statistically correlated from one block to the next, it is compressed using DPCM encoding. Only the smaller amount of difference between each DC value and the value of the DC coefficient in the block to its left needs to be represented in the final bit stream. This frequency conversion performed by applying the DCT provides a statistical decorrelation function to efficiently concentrate the signal into fewer high-amplitude values prior to applying quantization. These transform coefficients are then quantized.



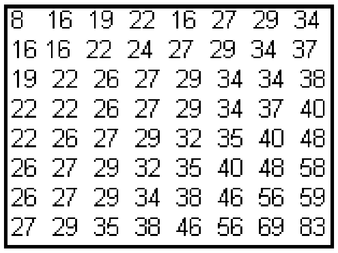
|  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|
| 255 | 255 | 255 | 255 | 255 | 255 | 255 | 255 |
| 255 | 187 | 204 | 255 | 255 | 255 | 255 | 255 |
| 255 | 122 | 20 | 102 | 230 | 255 | 255 | 255 |
| 255 | 153 | 0 | 0 | 35 | 136 | 213 | 255 |
| 255 | 196 | 0 | 0 | 0 | 0 | 17 | 94 |
| 255 | 247 | 43 | 0 | 0 | 0 | 0 | 0 |
| 255 | 255 | 82 | 0 | 0 | 0 | 0 | 0 |
| 255 | 255 | 128 | 0 | 0 | 0 | 0 | 0 |

|  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|
| 276 | 59 | 89 | 39 | 7 | -13 | -12 | -7 |
| 137 | -94 | -35 | 4 | 17 | 16 | 7 | 2 |
| 51 | 25 | -42 | -20 | -14 | 1 | 5 | 7 |
| -12 | 40 | -8 | -16 | -4 | -4 | -5 | -5 |
| -8 | 3 | 17 | -13 | -4 | 0 | 2 | -1 |
| 2 | 14 | 14 | 5 | -7 | 0 | -1 | 0 |
| -1 | -3 | -2 | 12 | 0 | -4 | -2 | 1 |
| -6 | 2 | -6 | 6 | 8 | -5 | -1 | 0 |

**Figure 3.5:** Example of 8 x 8 DCT

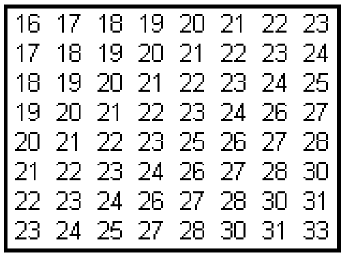
### Quantization

Quantization of digital data is the process of reducing the accuracy of a signal, by dividing it into some larger step size as finding the nearest multiple, and discarding the remainder/modulus. A **quantization matrix** is a string of 64-numbers (0-255) which tells the encoder how relatively important or unimportant each piece of visual information is. Each number in the matrix corresponds to a certain frequency component of the video image. The function of quantization matrix is to quantize high frequencies with coarser quantization steps that will suppress high frequencies with no subjective degradation, thus taking advantage of human visual perception characteristics. Quantization is performed by taking each of the 64 frequency values of the DCT block, dividing them by the frame-level quantizer, and then dividing them by their corresponding values in the quantization matrix. Finally, the result is rounded down. This significantly reduces, or completely eliminates, the information in some frequency components of the picture. Typically, high frequency information is less visually important, and so high frequencies are much more strongly quantized (highly reduced). The bits saved for coding high frequencies are used for lower frequencies to obtain better subjective coded images. There are two quantizer weighting matrices in Test Model (TM5) and intraquantizer weighting matrix and a nonintraquantizer weighting matrix; the latter is flatter since the energy of coefficients in interframe coding is more uniformly distributed than in intraframe coding. Quantization eliminates a large amount of data, and is the main lossy processing step in MPEG-1 video encoding.



|  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|
| 8 | 16 | 19 | 22 | 16 | 27 | 29 | 34 |
| 16 | 16 | 22 | 24 | 27 | 29 | 34 | 37 |
| 19 | 22 | 26 | 27 | 29 | 34 | 34 | 38 |
| 22 | 22 | 26 | 27 | 29 | 34 | 37 | 40 |
| 22 | 26 | 27 | 29 | 32 | 35 | 40 | 48 |
| 26 | 27 | 29 | 32 | 35 | 40 | 48 | 58 |
| 26 | 27 | 29 | 34 | 38 | 46 | 56 | 59 |
| 27 | 29 | 35 | 38 | 46 | 56 | 69 | 83 |

Intra quantizer weighting matrix



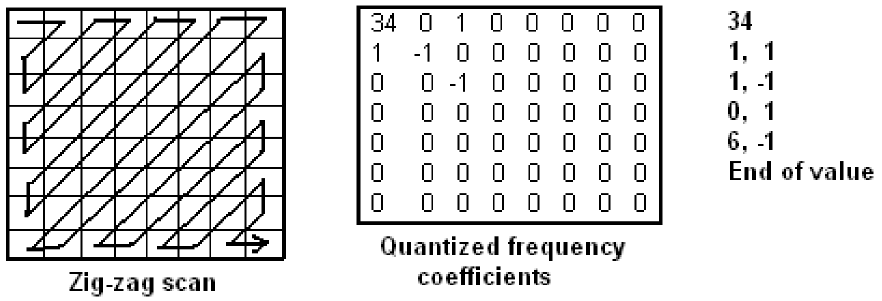
|  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|
| 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 |
| 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |
| 19 | 20 | 21 | 22 | 23 | 24 | 26 | 27 |
| 20 | 21 | 22 | 23 | 25 | 26 | 27 | 28 |
| 21 | 22 | 23 | 24 | 26 | 27 | 28 | 30 |
| 22 | 23 | 24 | 26 | 27 | 28 | 30 | 31 |
| 23 | 24 | 25 | 27 | 28 | 30 | 31 | 33 |

Nonintra quantizer weighting matrix

**Figure 3.6:** Quantization matrices

### Zig-zag Scan

The coefficients are processed in zig-zag order since the most energy is usually concentrated in the lower-order coefficients. The DCT block tends to have the most important frequencies towards the top left corner. The coefficients tend to zero towards the bottom-right. Maximum compression can be achieved by a Zig-zag scanning of the DCT block starting from the top left.



| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 34 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | -1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | -1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**34**
**1, 1**
**1, -1**
**0, 1**
**6, -1**
**End of value**

**Figure 3.7:** Zigzag scans to get pairs of zero-runs and value.

### Run-Length Encoder

The Zig-zag ordering of elements in an 8x8 matrix allows for a more efficient run-length coder. This is illustrated in Figure 3.7; with the Zig-zag order the run-length coder converts the quantized frequency coefficients to pairs of zero runs and nonzero coefficients.

34 0 1 0 –1 1 0 0 0 0 0 0 –1 0 0 0 0…….

After parsing, the pairs of zero runs and values are obtained:

34 | 0 1 | 0 –1 | 1 | 0 0 0 0 0 0 –1 | 0 0 0 0…….

These pairs of runs and values are then coded by a Huffman-type entropy coder for example for the above run/value pairs can be shown as in Table 3.2:

**Table 3.2: Huffman- type entropy encoder**

| **Run/Values** | **VLC(Variable Length Code)** |
|---|---|
| 34 | 0110 |
| 1, 1 | 0111 |
| 1,-1 | 110 |
| 0,1 | 0001011 |
| 6,-1 | 10 |
| End of block | |

**Huffman Coding**

This is a very popular method of entropy coding, and used in MPEG-1 video to reduce the data size. The data is analyzed to find strings that repeat often. Those strings are then put into a special table; with the most frequently repeating data assigned the shortest code. This keeps the data as small as possible with this form of compression. Once the table is constructed, those strings in the data are replaced with their (much smaller) codes, which reference the appropriate entry in the table. The decoder simply reverses this process to produce the original data.

## Decoding Process:

The decoding process is an inverse procedure of encoding. The block diagram of a typical decoder is shown in Figure 3.8.

The variable-length decoder (VLD) first decodes the coded data or video bit stream. This process yields the quantized DCT coefficients and motion vector data for each macroblock. The coefficients are inversely scanned and dequantized. The decoded DCT coefficients are then inverse-transformed to obtain the spatial-domain pixels. If the macroblock was intracoded, these pixels represent the reconstructed values without any further processing. However, if the macroblock is intercoded, then motion compensation is performed to add the prediction from the corresponding reference frame or frames.

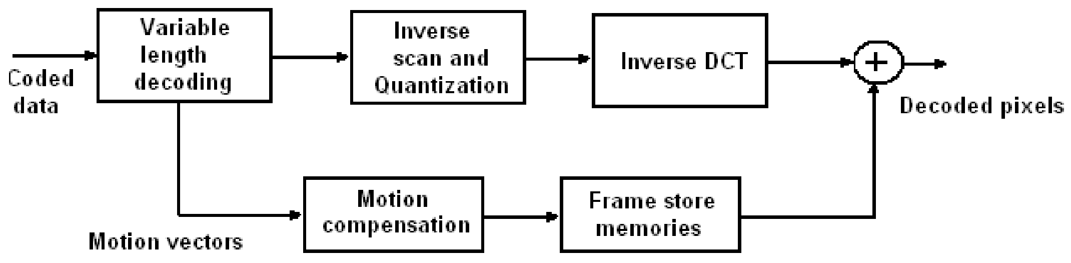


**Figure 3.8:** MPEG Video Decoder

## 3.4 Summary

This chapter has provided an overview of MPEG, video compression standard with detail of all codec steps and their effect on final bit stream. The next chapter will provide the literature survey that has been carried out for this research work.

# Chapter 4

# Literature Survey

This chapter will give an overview of the previous work done in the domain of this research work. In the past, different approaches have been proposed to address major features and issues related to a camera sensor network; these are briefly discussed in this chapter.

## 4.1 Related Work

A multi-camera sensor network consists of tiny fully equipped computer systems each capturing the scene from a particular angle. All these nodes have to transmit their findings to a central or common receiver. The available link capacity is not sufficient to carry this huge load. On the other hand each node's findings are important and play a significant role in formulation of results. So there should be some compression algorithm that should exploit spatial and temporal redundancies with in each video sequence as well as inter-sequence statistical redundancy should also be removed to acquire best compression ratio.

Researchers have worked for creating number of algorithms to get rid of this data repetition. Power and energy are the major constraints for a multi-sensor network. To preserve a sensor node energy and power and to save the available link capacity and to enhance node ability, inter-node communication is normally minimized. Numbers of ideas have been presented to cater these issues of a multi-terminal scenario. Some techniques allow minimum inter-node communication and some do not. Number of techniques has been implemented to get inter-node correlation. Like the concept of epipolar geometry, which has been used to explore video cameras geometry and some techniques even don't bother to remove such inter-camera sequence redundancy.

**Anshul Sehgal, Ashish Jagmohan, and Narendra Ahuja**, fellow IEEE in their paper titled as "Whyner-Ziv Coding of Video: An Error-Resilient Compression Framework" have used the concept of DSC [1]. This and similar papers [2],[3] have explained the work on the same line as utilizing the concept of DSC only for the exploitation of temporal correlation in a single video stream, but not exploiting the inter-sequence redundancy. Their approach leads to a low complexity encoder and better error resilience. They have focused on prevention of error propagation which is achieved by periodically transmitting a small amount of additional information termed as coset information to the decoder. They have defined “peg frames” and their associated epochs in a similar way to the corresponding quantities. A block diagram of proposed coset encoding algorithm is depicted in Figure 4.1.

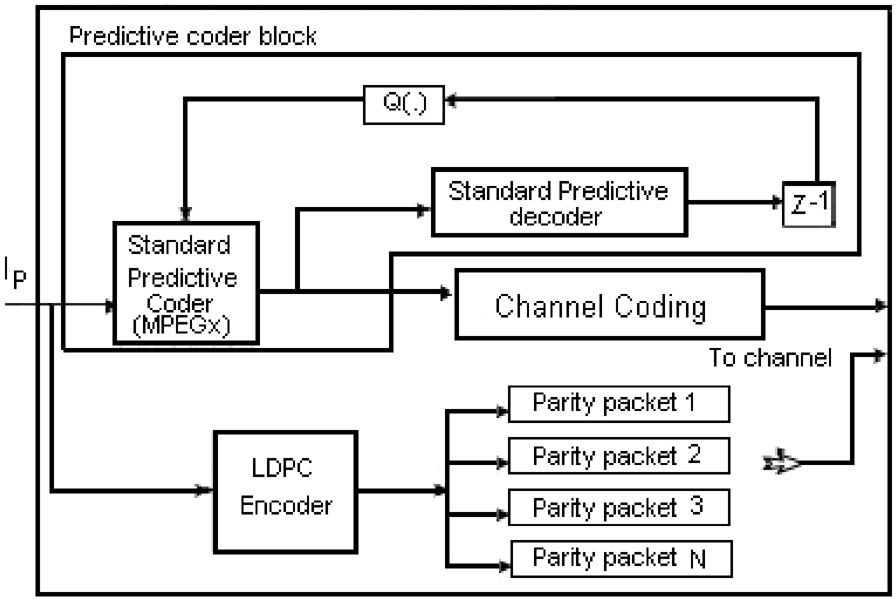


**Figure 4.1:** Block diagram of proposed video.

Video frame q(Ip) is obtained by applying the H.26L, forward transform to each 4x4 block of Ip. The resultant transform domain coefficients are quantized using the H.26L dead-zone quantizer. Coset information for the peg frame Ip is generated by applying LDPC coset codes to the transform-domain coefficients of the required peg frame q(Ip). During the decoding process all nonpeg-frames are reconstructed using a H.26L decoder. The decoder reconstructs the peg frames Ip by adding the displaced-frame difference Pp to Ip-1.

**Nicolas Gehrig and Pier Luigi Dragotti** from Communications and Signal Processing Group, Electrical and Electronic Engineering Department Imperial College

London, presented a paper titled as “DIFFERENT - Distributed and Fully Flexible Image Encoders for Camera Sensor Networks.” They have developed a distributed image coding technique for a multi-camera setup [4]. Certain assumptions were assumed about the camera location like placement of cameras in a horizontal line, and the objects are in a certain known range from the cameras. Their camera sensor network scenario is shown in Figure 4.2.

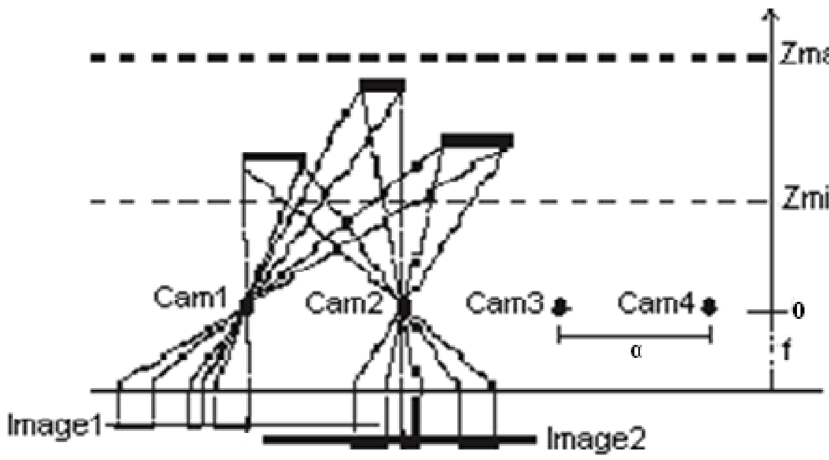


**Figure 4.2:** Camera sensor network scenario

They derived a lower bound on the minimum number of cameras required to perfectly reconstruct a scene. In this approach image encoder based on tree-structured algorithm can be modified. Generally correlation in the visual information is exploited using some geometrical information.

**Yang Yang,Vladimir Stankovi ´c, Wei Zhao, and Zixiang Xiong** from Dept of Electrical and Computer Engineering, Texas A&M University, in their paper titled as "Multiterminal Video Coding", addresses the concept of multi-terminal video coding in its most simple sense [5].This approach examines multi-terminal source coding of two correlated video sequences to save the sum rate over independent coding. Specifically, the first video sequence is coded by H.264 and used at the joint decoder to facilitate Wyner-Ziv coding of the second video sequence. The first I-frame of the right sequence is successively coded by H.264 and Slepian-Wolf coding. An efficient stereo matching algorithm is then adopted at the decoder to produce pixel-level disparity maps between the corresponding frames of the two decoded video sequences. Based on the disparity maps, side information for both motion vectors and motion-compensated residual frames of the second sequence are generated at the decoder before Wyner-Ziv encoding. Technique used for finding corresponding points require computationally involved depth estimation where beside horizontal disparities, vertical disparities among video frames are also exploited. Multi-terminal video encoder-decoder architecture for right I-frame is shown in Figure 4.3.

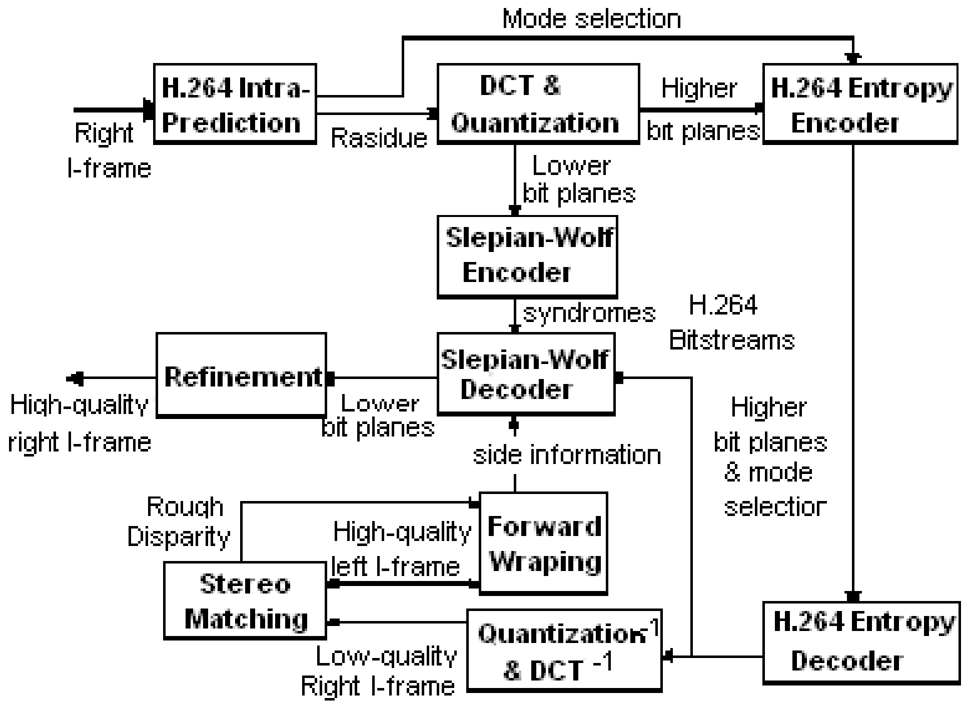


**Figure 4.3:** Multi-terminal video encoder-decoder (right I-frame)

**Bi Song, Eterm Tuncel, Amit K. Roy-Chowdhury** from Department of Electrical Engineering, University of California, Riverside have proposed a paper titled as "Towards A Multi-Terminal Video Compression Algorithm By Integrating Distributed Source Coding With Geometrical Constraints." They have presented the approach of epipolar lines between two frames macroblocks [6]. Their algorithm comprises two parts, one part depicts motion estimation in distributed fashion which yields corresponding macroblocks in two provided images or two sequence frames, and other part is for distributed coding of these corresponding macroblocks. While the portion of the two video frames carrying non-overlapping macroblocks is coded by conventional video coding method. This lossy compression scheme was actually about integration of video analysis tool with concept of distributed source coding. The working mechanism of distributed motion estimation part is shown in Figure 4.4.

The epipolar geometry toolbox has also been developed in MATLAB that works for both panoramic and pinhole cameras [7]. This toolbox has been created to provide MATLAB user with an extensible framework for the creation and visualization of multi-camera scenario as well as for the manipulation of the visual information and the geometry between them. Functions provided for vision sensors include camera placement and visualization, computation, estimation of epipolar geometry entities and many others.

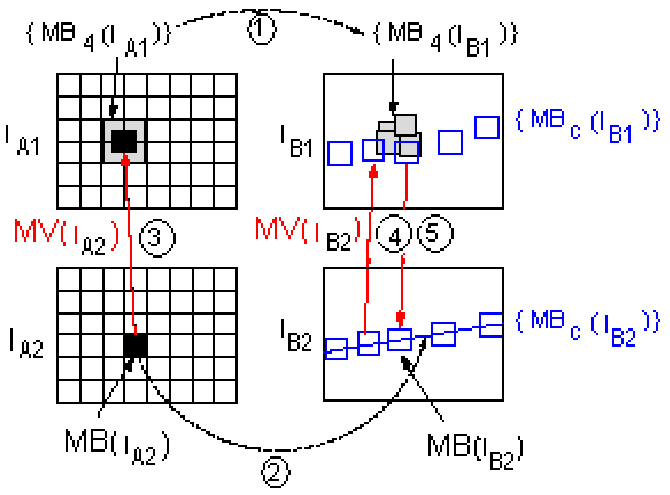


**Figure 4.4:** Pictorial description of the proposed correspondence tracking algorithm. (The numbers in circles indicate the steps of the algorithm.)

Two similar papers by the same authors have also been recently proposed. One is the just similar approach that is implementation of DSC in multi-camera environment using epipolar geometry [8]. And the other one includes model based tracking as:

**Bi Song, Eterm Tuncel, Amit K. Roy-Chowdhury** in their paper titled as "A Multi-Terminal Model-Based video Compression Algorithm." present a novel 3D model-based distributed video coding algorithm [9]. It is based on independent, model-based tracking of multiple sources and distributed coding of the tracked feature points. The model-based tracking scheme provides correspondence between the overlapping set of features that are visible in the different views. While the motion estimates obtained from the tracking algorithm remove temporal redundancy and the 3D model accounts for removing spatial redundancy, distributed coding is used to eliminate inter-sensor redundancy. Model-based tracking algorithm comprises the following steps:

Step E1: Estimate the motion from $I_t$ to $I_{t+1}$ using the tracking algorithm. Render the image, $\bar{I}_{t+1}$, using the estimated motion and 3D model.

Step E2: Compute the residual $\delta I_{t+1} = I_{t+1} - \bar{I}_{t+1}$.

If the residual is above a certain threshold, go to Intra-frame coding.

Else proceed to next step.

Step E3: Quantize the result from Step E2.

Step E4: Transmit the quantized residuals, along with the quantized rotation and translation vectors.

The following are the steps taken by the decoders.

Step D1: Dequantize the 3D motion estimates and illumination parameters.

Step D2: Dequantize the residuals, and denote it as $\delta \hat{I}_{t+1}$.

Step D3: Using the 3D motion estimates and the 3D model, synthesize an estimate of the rendered image $\hat{\bar{I}}_{t+1}$.

Step D4: Obtain the reconstructed image as $\hat{I}_{t+1} = \hat{\bar{I}}_{t+1} + \delta \hat{I}_{t+1}$

**Markus Flierl and Bernd Girod** from Max Planck Center for Visual Computing and Communication Stanford University, California, in their paper titled as "Coding of Multi-View Image Sequences with Video Sensors." have used the concept of side information at decoder [10]. They have arranged the video sensors in an array to view the same scene from different angles. These video sensors process highly correlated information; such correlation is exploited by centralized disparity compensation at the decoder. The central decoder is designed to perform disparity estimation on previously decoded images of the multi-view image sequences. Basic architecture of disparity compensation at central decoder is shown in Figure 4.5.

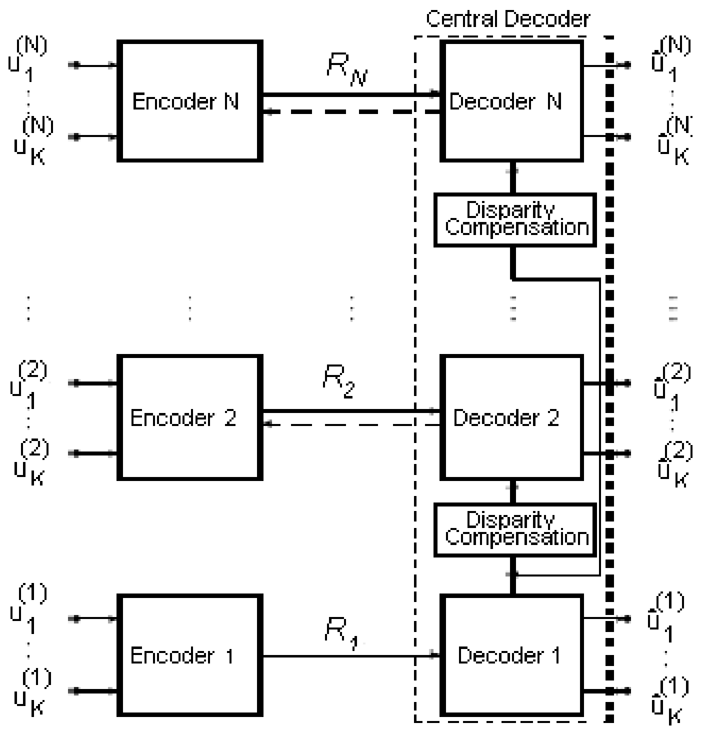


**Figure 4.5:** Distributed coding scheme with disparity compensation at the central decoder.

N multi-view image sequences are represented by u(k) ^ n with k = 1,2 …,K temporally successive frames of n = 1,2,…,N views. The encoding scheme comprises N encoders that operate independently as well as one central decoder. The latter is made up of N-1 "Wyner-Ziv" decoder n=2,…,N that are dependent on decoder 1. The side information for decoder n with n=2,….N can be improved by performing disparity compensation. As the video signals are not stationary, decoder n with n=2,….,N is decoding with feed-back.

## 4.2 Summary

This research work is an extension of a video compression standard (MPEG), which has been designed to cater various issues in a camera sensor network as well as keeping the communication among the sensors at minimum. This chapter has focused on various technologies and techniques that have been developed to address requirement issues of a multi camera setup. Next chapter will focus on implementation details of proposed scheme to make it more understandable.

# Chapter 5

# Proposed Methodology

This chapter gives the details of proposed system architecture and its work methodology. The chapter explains inter-node communication strategies, followed by system architecture and system flow chart. Additionally snapshots of system Graphical User Interface have also been provided for better understanding.

## 5.1 Proposed Scheme

The proposed scheme is a lossy compression scheme for a camera-sensor network removing spatial and temporal redundancies (with in each video sequence), and inter-sequence statistical redundancy (among video sequences) as well as keeping communication among the sensor nodes at minimum. Our proposed multi-camera setup consists of number of camera sensor nodes arbitrarily located in space, capturing the same scene from different angles depending upon their locations. All cameras are calibrated with some initial time t = 0. Algorithm is initiated by picking randomly any node. Camera sensor nodes are allowed to communicate in two different ways, termed as Strategy-1 and Strategy-2. Strategy-1 is shown in Figure 5.1.

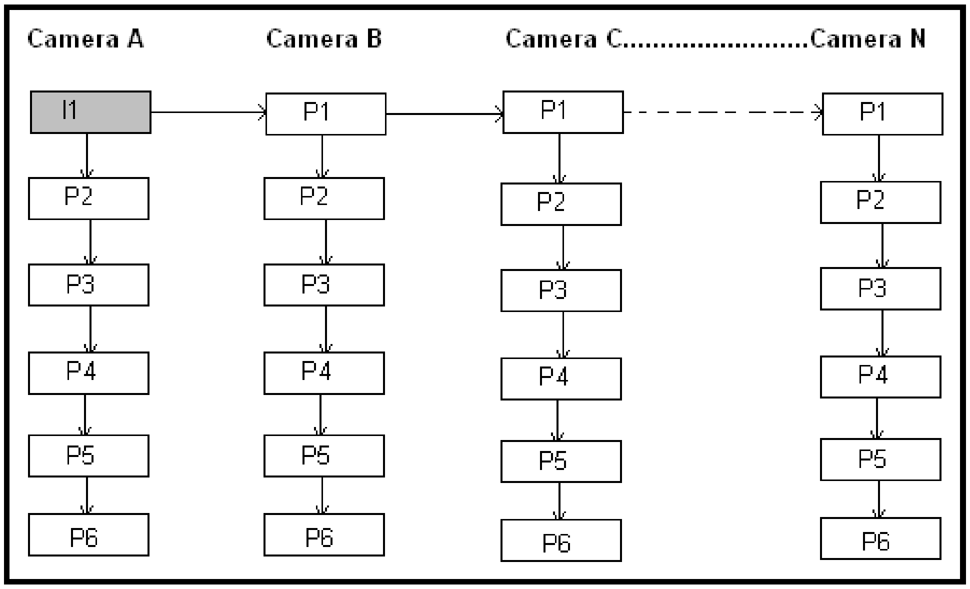


**Figure 5.1:** Frame Construction Strategy-1

In this strategy, after random selection of first camera nodes, its GOP is generated and first frame of that node video sequence is taken as reference I-frame and is intracoded**.**

As well as this first I-reference frame is used to encode the whole GOP (generated previously) of this camera node. Afterwards, this I-reference frame is to be passed to a nearest neighbor node and now the first frame of this next node video sequence is to be passed to next neighbor and the whole process is repeated for this node too. After generating the video sequence, first frame of this video sequence undergoes motion compensation using previous node I-reference frame while remaining frames of this node are intercoded using this first and then following P-frames of the video sequence. After doing some pre-processing, each encoder encodes its GOP (5-6 frames) using MPEG encoder and yields a compressed bit stream. Second strategy for camera nodes communication is shown in Figure 5.2.

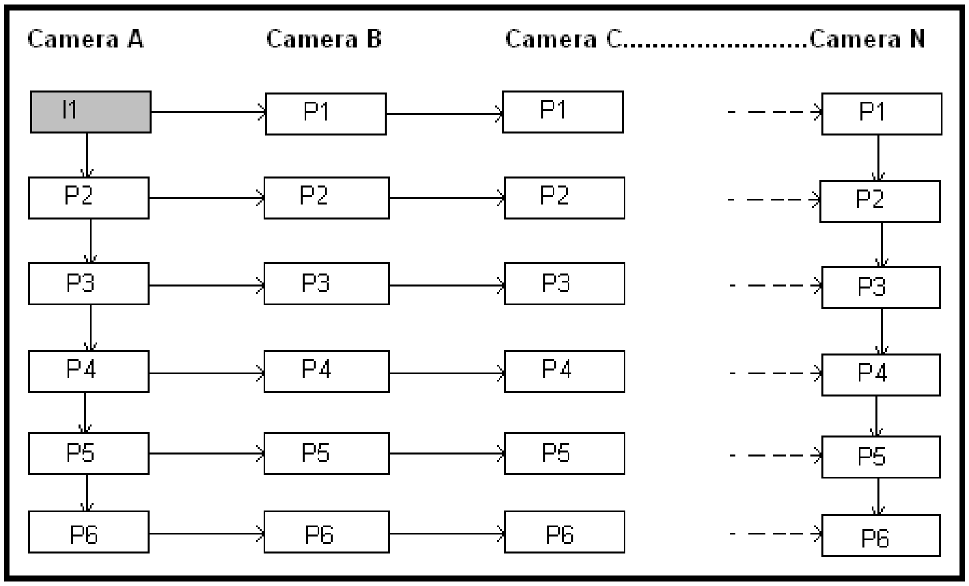


**Figure 5.2:** Frame Construction Strategy-2

In this strategy, after random selection of the camera node, its GOP is generated and first I-reference frame is intracoded as well as used for encoding of its whole GOP, and it is passed to the nearest neighbor. Now the condition is a little bit different, this time this I-frame is used just to do motion estimation and compensation for the first frame of this second camera node GOP, and remaining frames of the GOP are encoded using their previous neighbor parallel P-frames. As P22 (node-2 P2-frame) is compensated using P2 (node-1 P2 frame) frame and vise versa. For both the strategies, each sensor node is deployed with its own encoding algorithm provided that I-frame is to be transmitted periodically from one node to the next one to keep the algorithm in flow.

System decoder is a joint decoder carrying all sensor nodes video bit streams. Decoding is done by exploiting individual motion vector sets and respective decoded I-frame. After implementing standard MPEG decoding steps, individual video sequences are reconstructed. Depending upon number of camera nodes, for one I-frame, number of P-frames in resultant GOP is derived as:

Total Number of P-frames = number of sensor nodes * 6 frames (for each sensor node)

For example for 3 camera sensor nodes network overall GOP consists of 17 P-frames and one I-frame, so saving the resultant required bandwidth for this bit stream transmission up to a greater extent. This above approach is designed to remove inter-sequence statistical redundancy; while idea of motion-compensation is implemented on each individual node using that one I-frame, removes the temporal redundancy in each individual video sequence. And standard encoding steps like application of discrete cosine transform is sufficient to remove spatial redundancy with in each individual video frame.

In both the communication strategies, system is capable of switching in two working modes, termed as Scenario- A and Scenario-B. In Scenario-A system reconstructs the videos with comparatively better values of PSNR, while less compression ratio and compression percentage. While for Scenario-B, system response time is less compression ratio as well as compression percentage saving is better but decoded video quality is comparatively degraded. Both working modes can be implemented in different environments with different set of requirements.

A MATLAB toolbox has been developed for MPEG extension in multi-camera scenario. This toolbox consists of number of m-files. Each m-file is specialized for a specific step of encoding-decoding process. Moreover, a user interface has been designed for flexible user interaction with the execution of a particular multi-camera scenario, and analysis of results in two working modes of the system under two different communication strategies.

## 5.2 Proposed Scheme Architecture

Proposed scheme can be defined architecturally as in Figure 5.3.

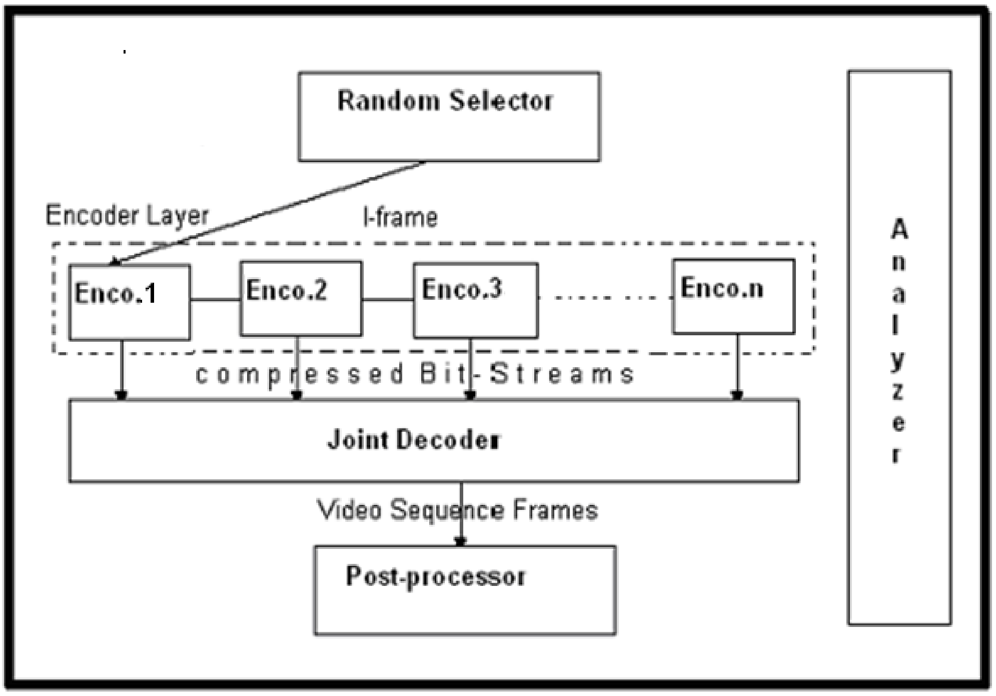


**Figure 5.3:** Proposed Scheme Architecture

Architecture components can be defined as:

### 5.2.1 Random Selector

This module is designed to randomly pick any sensor node with in the camera sensor network for the initiation of proposed algorithm. Then encoding process starts in that sensor node using standard MPEG encoding steps. Finally this node transmits its I-frame as reference frame to its nearest neighbor for continuity of algorithm execution.

### 5.2.2 Encoder Layer

Encoder layer comprises a number of encoders depending upon number of camera sensor nodes. Each encoder has its own encoding algorithm implemented. Through out the encoder layer, reference I-frames are transmitted from one node to the next one. Finally compressed bit streams are sent to channel for transmission.

### 5.2.3 Joint Decoder

Since all encoders send their compressed findings to a central decoder. Here standard MPEG decoding process takes place and video sequence frames are the output.

### 5.2.4 Post-processor

Each video sequence pictures are finally fed to post-processor which is logically designed to construct the individual video sequences.

### 5.2.5 Analyzer

Analyzer is a module which runs in parallel to this algorithm and specialized for capturing certain resultant parameters that are used for result analysis.

Multi-camera scenario is implemented by implanting two digital cameras capturing two side views of an object. Video sequences are being captured by these two sensors. Encoding is carried out after implementation of motion compensation between I-reference frames of the two video sequences. Then the two video sequences are encoded independently using standard MPEG encoding procedure.

## 5.3 System Flow Chart

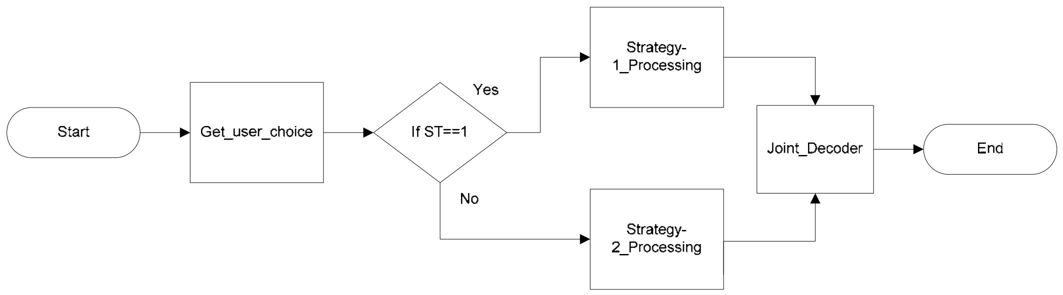


**Figure 5.4:** System Flow Chart (Abstract)

In next diagram, modules Strategy-1_Processing and Strategy-2_Processing are explained as:

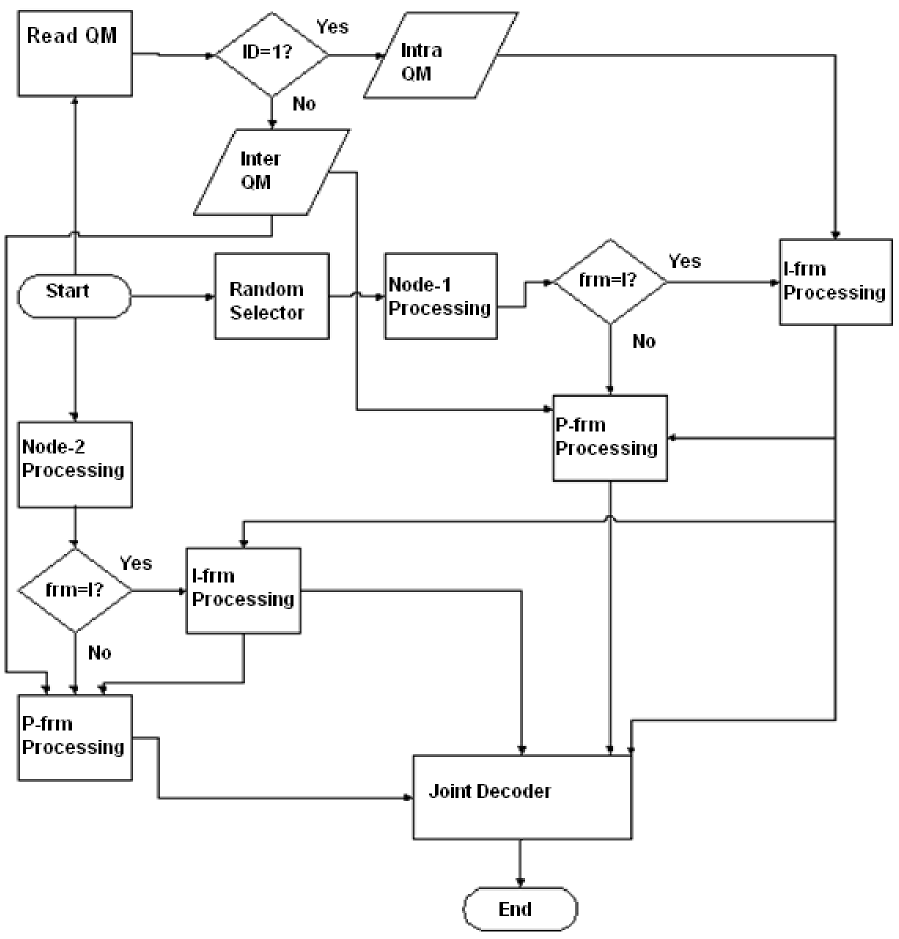


**Figure 5.5:** Strategy-1 Description

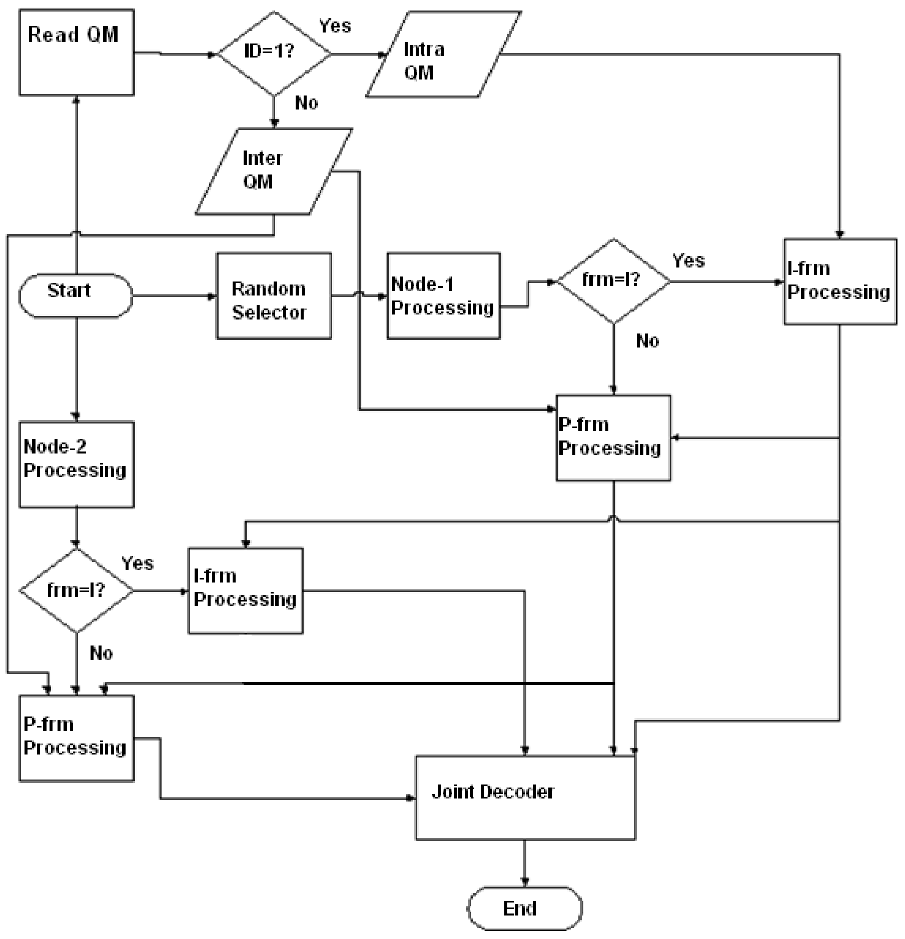


**Figure 5.6:** Strategy-2 Description

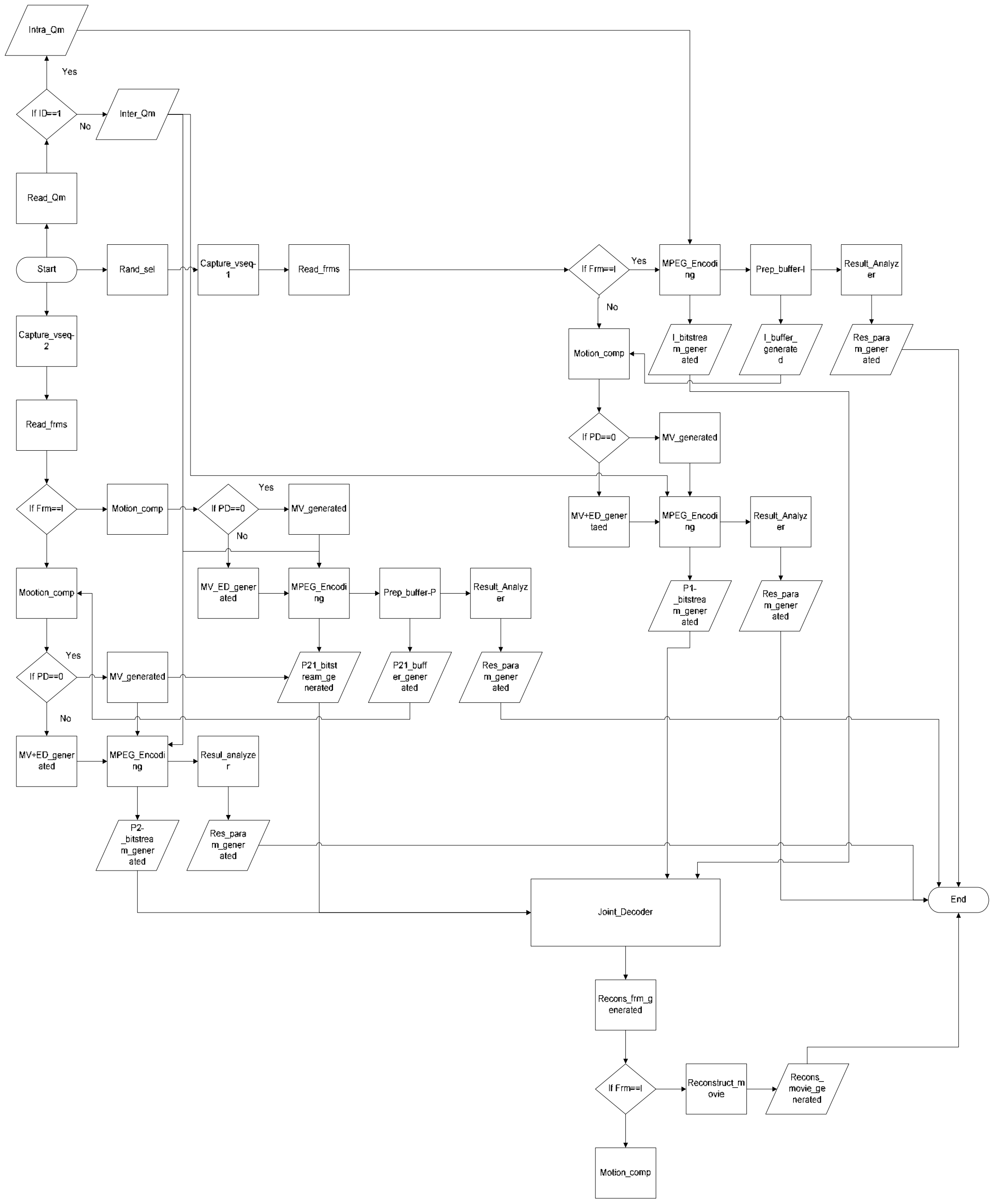
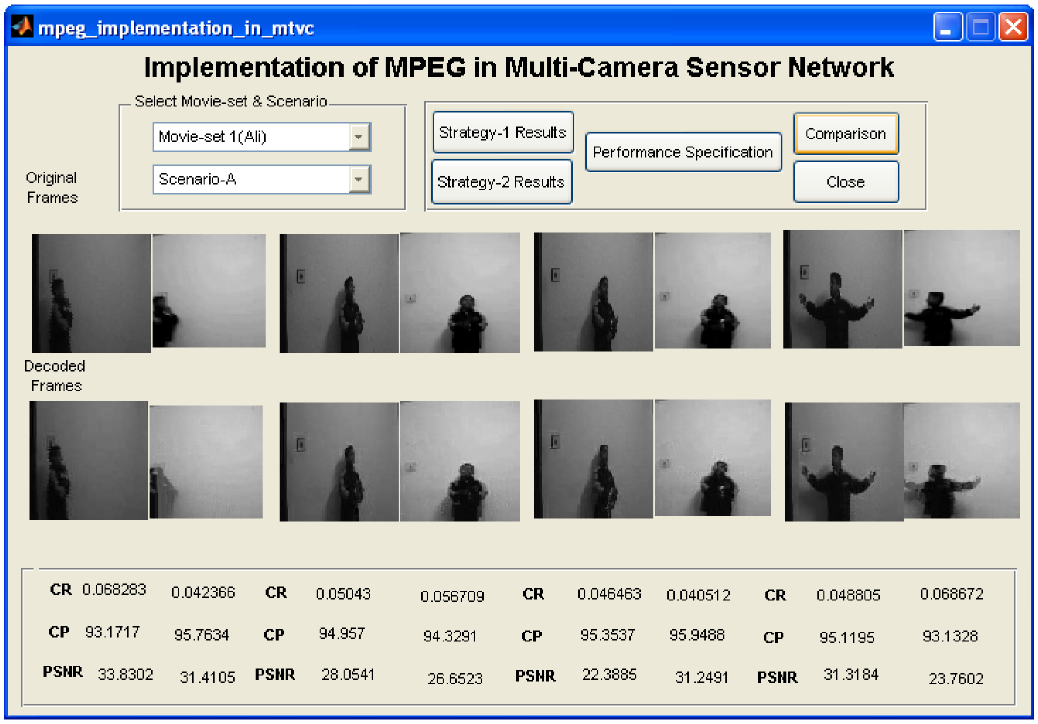


**Figure 5.7:** System Flow Chart (Full)

## 5.4 User Interfaces

A user interface has been designed for easy and flexible user interaction with proposed system. First GUI (Graphical User Interface) is provided with options of two communication strategies. User can select any of system working mode and a particular movie-set as well. Then user can see compression results for any of the communication strategy. Original and decoded frames of the respective movie set are displayed along with the values of performance parameters as Peak Signal to Noise Ratio (PSNR), Compression Ratio (CR) and Compression Percentage (CP).


mpeg_implementation_in_mtvc
Implementation of MPEG in Multi-Camera Sensor Network
Select Movie-set & Scenario
Movie-set 1(Ali)
Scenario-A
Strategy-1 Results
Strategy-2 Results
Performance Specification
Comparison
Close
Original Frames
Decoded Frames

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| CR | 0.068283 | 0.042366 | CR | 0.05043 | 0.056709 | CR | 0.046463 | 0.040512 | CR | 0.048805 | 0.068672 |
| CP | 93.1717 | 95.7634 | CP | 94.957 | 94.3291 | CP | 95.3537 | 95.9488 | CP | 95.1195 | 93.1328 |
| PSNR | 33.8302 | 31.4105 | PSNR | 28.0541 | 26.6523 | PSNR | 22.3885 | 31.2491 | PSNR | 31.3184 | 23.7602 |

**Figure 5.8:** First System GUI

For any of the communication strategies selected, user can see the results comparison for both system working modes by clicking on ‘Comparison’ button. Next GUI frame is again provided with three buttons for CR, CP and PSNR comparison for selected movie set as shown in Figure 5.9(a), (b) and(c) respectively.

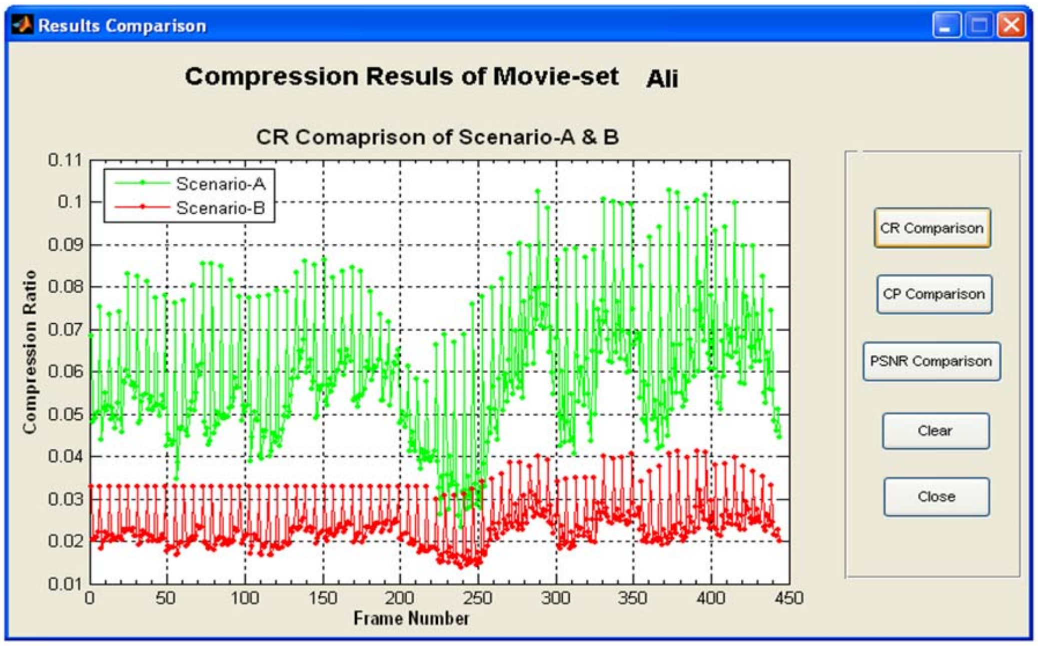


**Figure 5.9(a):** Results Comparison GUI showing CR Comparison

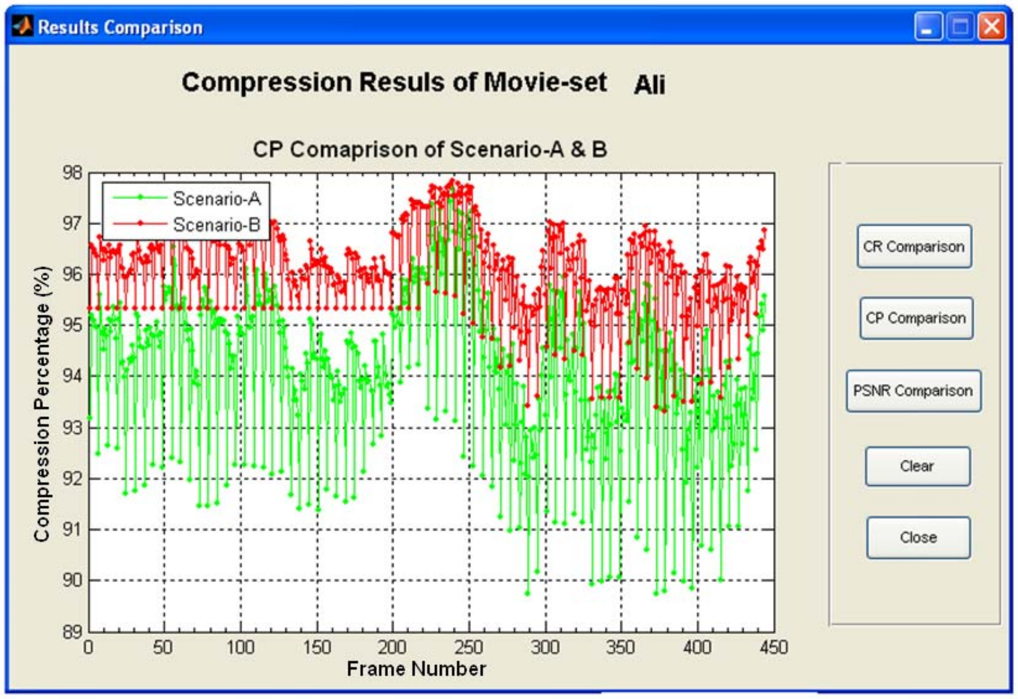


**Figure 5.9(b):** Results Comparison GUI showing CP Comparison

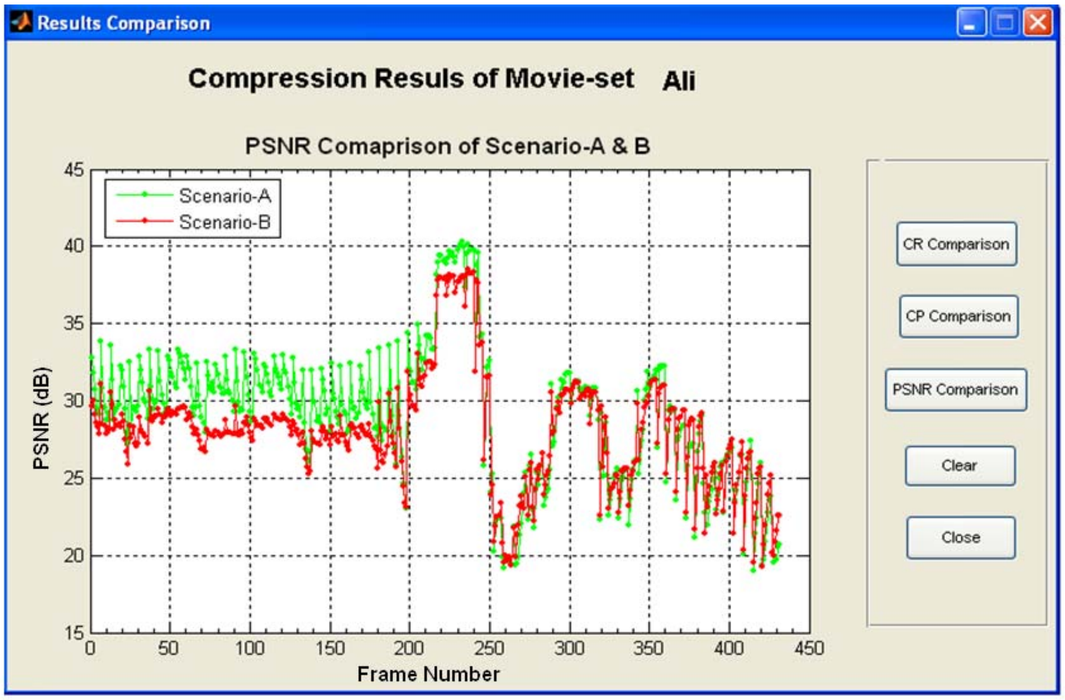


**Figure 5.9(c):** Results Comparison GUI showing PSNR Comparison

## 5.5 Summary

In this chapter, system implementation has been discussed with the help of respective figures. Different steps involved in algorithm execution and results evaluation have been shown through the designed user interface. The next chapter is about system performance evaluation and results discussion.

Chapter 6

# Performance Evaluation and Results Analysis

In this chapter the results and evaluations of the proposed scheme have been discussed in detail.

**Table 6.1: System specifications**

| System Requirement | |
|---|---|
| RAM | 1GB |
| Processor | 1.37 Dual Core |
| **Software Requirement** | |
| Operating System | Windows XP |
| Tool Box | MATLAB 7.0 |
| Other Software Packages | Video Convert Master-<br>K-Lite full Codec |

## 6.1 Criteria for evaluations

Like any compression system, in this proposed research study the compression ratio and frame/video quality are of main concern. Various metrics can be used to measure the performance of a compression system. Following objective performance metrics have been analyzed to measure attained compression ratio and quality of a reconstructed video frames:

**(a)** Peak Signal to Noise Ratio (PSNR)

**(b)** Compression Ratio (CR)

**(c)** Compression Percentage (CP)

**(d)** Mean Squared Error (MSE)

### 6.1.1 Peak Signal to Noise Ratio (PSNR)

This objective metric is used to measure the quality of reconstructed video frame. More value of PSNR indicates better quality. It is most easily defined using the term MSE for two mxn video frames f and f'. Where f is considered as original and f' is reconstructed or approximation of the other. MSE is defined by the following formula as:

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

And the mathematical formula to find PSNR is given as:

$$PSNR = 10 \log_{10} (255*255/MSE)$$

### 6.1.2 Compression Ratio (CR)

This performance metric measures the compression achieved by compression system.

$$CR = f' / f$$

Where f′ and f are compressed frame bytes and original frame bytes respectively. Compression ratio i.e. 5 specifies that for every 1 unit in the compressed data set, there are 5 information carry units in original frame.

### 6.1.3 Compression Percentage (CP)

Percent compression achieved by the compression system can be calculated by using following formula:

$$CP = 100 - (f' / f) X 100$$

Again f′ is compressed frame bytes and f is original frame bytes.

### 6.1.4 Distortion (MSE)

The mean square error or MSE of a reconstructed video frame is a way to quantify the difference between it and the original video frame being reconstructed. MSE corresponds to the average of this difference, and is given by the formula:

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

Where f′ is the reconstructed video frame and f is the original video frame.

## 6.2 System Performance Evaluation

The two proposed communication strategies allow different level of communication among camera sensor nodes in a multi-camera network. There are certain aspects of system performance since it is capable of switching among different working modes depending upon a particular scenario and its requirements. When camera sensor nodes in a wireless network, are allowed to exchange their findings time to time then the quality of resultant decoded video frames will be different. And entirely different results are obtained in situation where communication among camera sensors nodes is kept at minimum. This camera sensor node communication and video quality trade off can be shown as following:

### 6.2.1 Camera Node Communication and Video Quality

This can be shown from the results of different movie sets, which are encoded using both the strategies. And results can be analyzed using different performance metrics like CR, CP and PSNR. Graphical and tabular results are shown as below.

**Table 6.2: Comparison of Compression Ratio (CR) achieved in both strategies**

| Frames in a GOP | CR-Strategy-1 | CR-Strategy-2 |
|---|---|---|
| **I1** | 0.032768 | 0.032768 |
| **P2** | 0.02386 | 0.02386 |
| **P3** | 0.020317 | 0.020317 |
| **P4** | 0.020836 | 0.020836 |
| **P5** | 0.020958 | 0.020958 |
| **P6** | 0.022171 | 0.022171 |
| **P21** | 0.029778 | 0.029778 |
| **P22** | 0.01799 | 0.029015 |
| **P23** | 0.015564 | 0.027832 |
| **P24** | 0.014809 | 0.029465 |
| **P25** | 0.016739 | 0.02887 |
| **P26** | 0.01519 | 0.029709 |

Above are the tabular representations of compression results of only one GOP of a movie set, following is the compression ratio plot of the two movie frames.

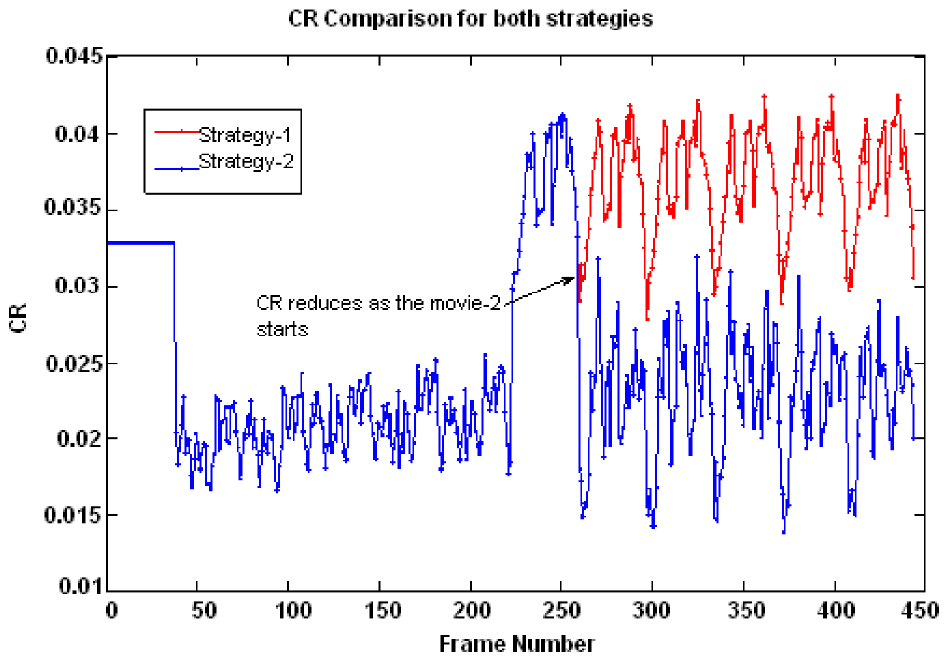


**Figure 6.1:** CR Comparison of both strategies

Next are the PSNR results of the two movie-set frames shown in graphical as well as tabular forms.

**Table 6.3: Comparison of Peak Signal to Noise Ratio (PSNR) in both strategies**

| **Frames in a GOP** | **PSNR-Strategy-1** | **PSNR-Strategy-2** |
|---|---|---|
| **I1** | 29.634 | 29.634 |
| **P2** | 30.024 | 30.024 |
| **P3** | 29.14 | 29.14 |
| **P4** | 28.577 | 28.577 |
| **P5** | 28.325 | 28.325 |
| **P6** | 27.882 | 27.882 |
| **P21** | 36.805 | 36.805 |
| **P22** | 37.768 | 37.324 |
| **P23** | 37.997 | 36.595 |
| **P24** | 37.996 | 35.641 |
| **P25** | 37.84 | 34.422 |

|  |  |  |
|---|---|---|
| **P26** | 37.932 | 34.095 |

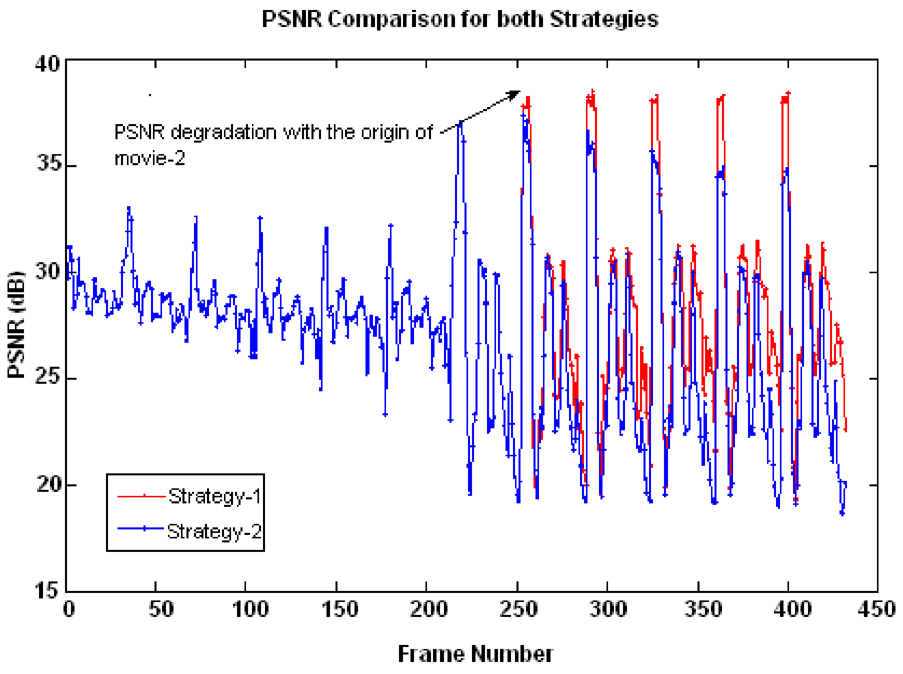


**Figure 6.2:** PSNR Comparison of both strategies

Next are the compression percentage (CP) results of the two movie-set frames shown in graphical as well as tabular forms.

**Table 6.4: Comparison of Compression Percentage (CP) achieved in both strategies**

| **Frames in a GOP** | **CP-Strategy-1** | **CP-Strategy-2** |
|---|---|---|
| **I1** | 95.338 | 95.338 |
| **P2** | 96.571 | 96.571 |
| **P3** | 96.496 | 96.496 |
| **P4** | 96.484 | 96.484 |
| **P5** | 96.391 | 96.391 |
| **P6** | 96.346 | 96.346 |
| **P21** | 95.801 | 95.801 |

|  |  |  |
|---|---|---|
| **P22** | 97.312 | 95.908 |
| **P23** | 97.619 | 95.972 |
| **P24** | 97.711 | 95.782 |
| **P25** | 97.505 | 95.797 |
| **P26** | 97.691 | 95.772 |

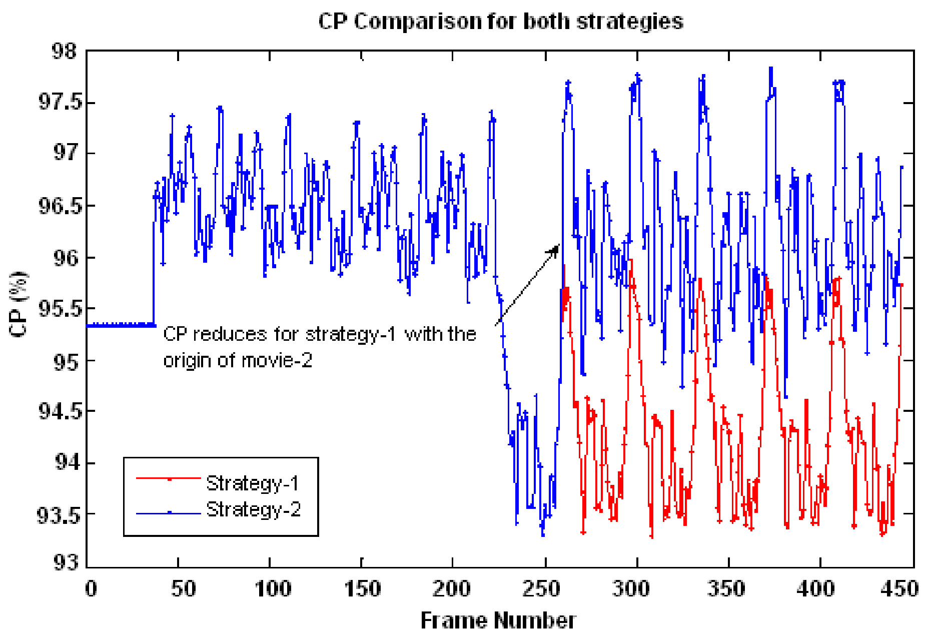


**Figure 6.3:** CP Comparison of both strategies

According to Figures 6.1, 6.2 and 6.3, it is clear that both the strategies start from the same points and give exactly same results up till movie-1. But with the beginning of second movie different results are obtained for all the three compression metrics like PSNR, CR and CP. strategy-1 gives good results on the basis of video quality i.e. good PSNR and strategy-2 gives good results on the basis of saving ratio.

### 6.2.2 System Working Mode Comparison

Critical results analysis highlights another system performance aspect that changing the response time of the system will strongly effects resultant video frame quality. This relationship can be clarified by system working modes comparison. System working mode-A (Scenario-A) can be applicable in the situation where its response time is not an issue but only better quality or fine information details are required. Like wise, when information results are required at some broader level (like for detection purposes) as well as quick system response is required, then it can switch in working mode-B (Scenario-B). In both the scenarios, system performance can be analyzed on the basis of values of compression metrics.

**Table 6.5: Comparison of Compression Ratio (CR) achieved in both Scenarios**

| **Frames in a GOP** | **CR-Scenario-A** | **CR-Scenario-B** |
|---|---|---|
| **I1** | 0.068283 | 0.032768 |
| **P2** | 0.048103 | 0.020386 |
| **P3** | 0.048752 | 0.020317 |
| **P4** | 0.049103 | 0.020836 |
| **P5** | 0.050217 | 0.020958 |
| **P6** | 0.050545 | 0.022171 |
| **P21** | 0.066353 | 0.029778 |
| **P22** | 0.038795 | 0.01799 |
| **P23** | 0.032616 | 0.015564 |
| **P24** | 0.026367 | 0.014809 |
| **P25** | 0.035172 | 0.016739 |
| **P26** | 0.030167 | 0.01519 |

Above are the tabular representations of compression results of only one GOP of a movie set, following is the compression ratio plot of the two movie frames.

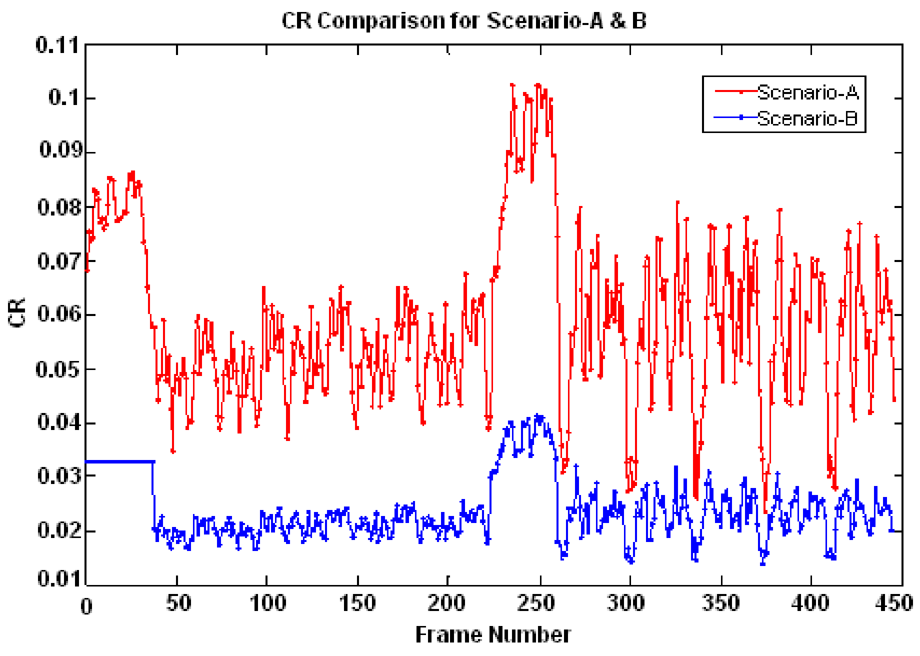


**Figure 6.4:** CR Comparison for Scenario A and B

Next is the PSNR comparison of two system working modes, shown in graphical as well as tabular forms.

**Table 6.6: Comparison of Peak Signal to Noise Ratio (PSNR) in both strategies**

| **Frames in a GOP** | **PSNR-Scenario-A** | **PSNR-Scenario-B** |
|---|---|---|
| **I1** | 32.769 | 29.634 |
| **P2** | 31.808 | 30.024 |
| **P3** | 30.701 | 29.14 |
| **P4** | 29.568 | 28.577 |
| **P5** | 29.375 | 28.325 |
| **P6** | 29.096 | 27.882 |
| **P21** | 38.153 | 36.805 |
| **P22** | 38.963 | 37.768 |
| **P23** | 39.372 | 37.997 |
| **P24** | 39.436 | 37.996 |
| **P25** | 39.012 | 37.84 |

|  |  |  |
|---|---|---|
| **P26** | 39.099 | 37.932 |

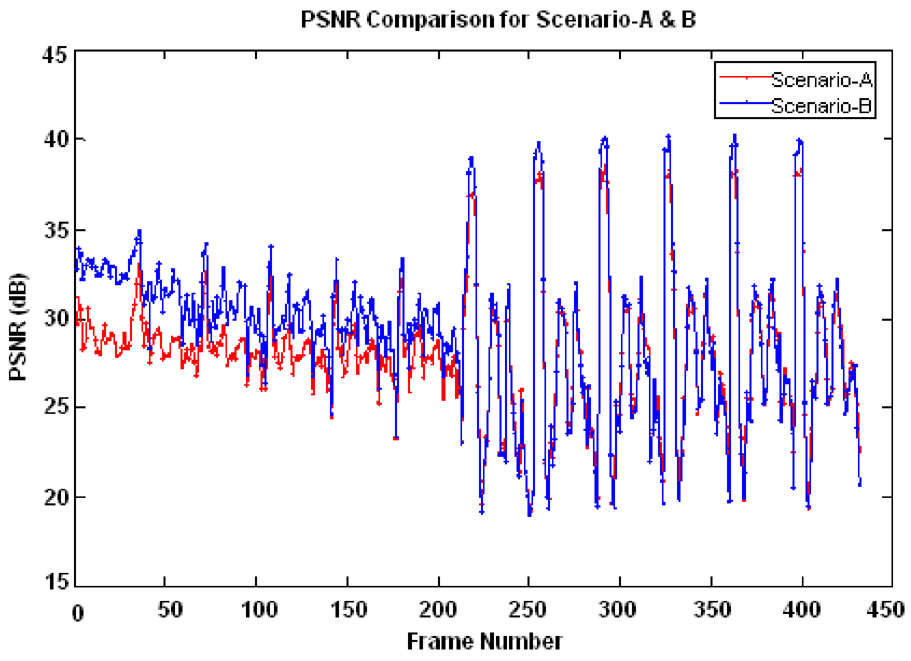


**Figure 6.5:** PSNR Comparison for Scenario A and B

Next is the CP comparison of two system working modes, shown in graphical as well as tabular forms.

**Table 6.7: Comparison of Compression Percentage (CP) achieved in both strategies**

| Frames in a GOP | CP-Scenario-A | CP-Scenario-B |
|---|---|---|
| **I1** | 93.172 | 95.338 |
| **P2** | 95.19 | 96.571 |
| **P3** | 95.125 | 96.496 |
| **P4** | 95.09 | 96.484 |
| **P5** | 94.978 | 96.391 |
| **P6** | 94.946 | 96.346 |

|  |  |  |
|---|---|---|
| **P21** | 93.365 | 95.801 |
| **P22** | 96.12 | 97.312 |
| **P23** | 96.738 | 97.619 |
| **P24** | 97.363 | 97.711 |
| **P25** | 96.483 | 97.505 |
| **P26** | 96.983 | 97.691 |

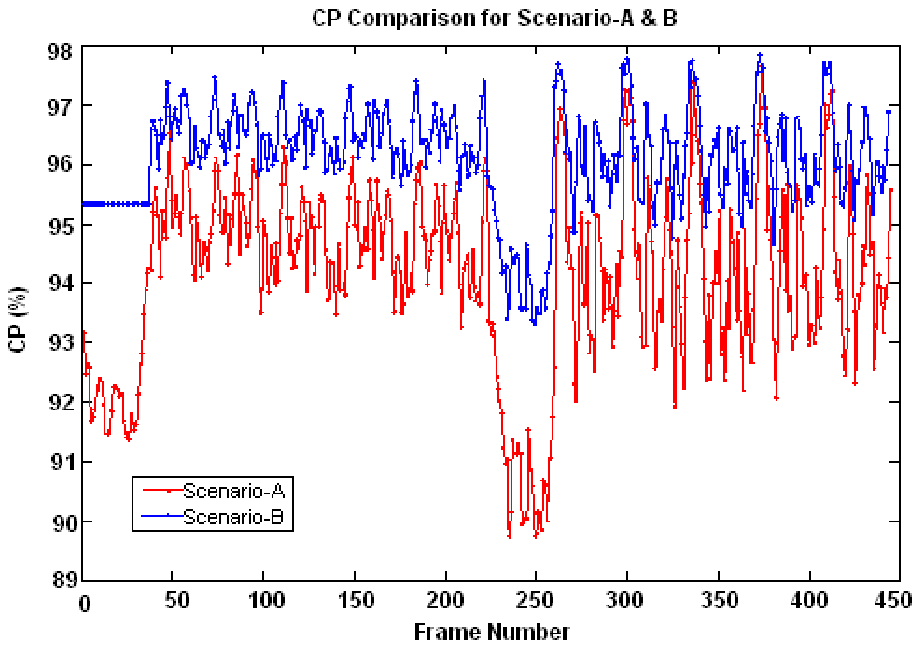


**Figure 6.6:** CP Comparison for Scenario A and B

According to Figures 6.4, 6.5 and 6.6 same point is a bit more clarified that, Scenario-A is better in terms of PSNR values but at the same time takes more processing time as well as give results with comparatively less saving ratio. While Scenario-B is good in terms of processing time and saving ratio but gives results with low PSNR.

### 6.2.3 System Performance against Various Compression Parameters

Moreover, to check system response at various bitrates in terms of different compression parameters, results are analyzed (Figures 6.7-6.12). System response is examined in both strategies under both working modes separately.

➢ **PSNR vs. Bitrate**

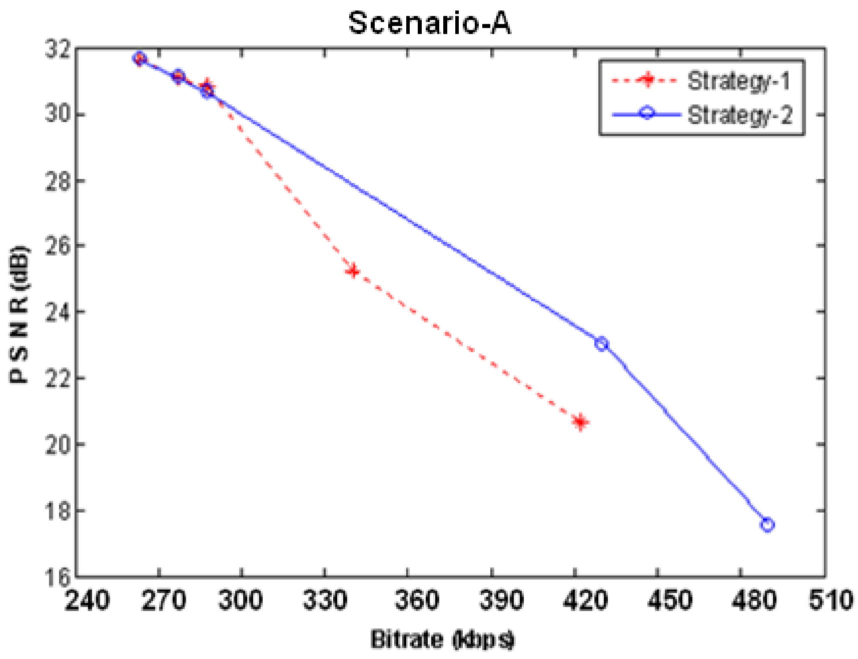


**Figure 6.7:** PSNR vs. Bitrates for Strategy-1 and 2 (Scenario-A)

This Figure shows that with the progress in number of movie frames for both the strategies, there is a fall in PSNR value as bitrate increases, as good PSNR is achieved at lower bitrates. Both the strategies start from the same point but with the beginning of movie-2 data, both the graphs are separated as strategy-1 ends at PSNR 20 dB with bitrate of 420 kbps. While strategy-2 ends with PSNR value 17 dB at bitrate 490 kbps. So for both the strategies quality of decoded video frame is found at lower bitrates for Scenario-A. Same relationship can be observed for Scenario-B as shown in Figure 6.8. One major difference for both the scenarios is that scenario-B ends at comparatively low bit rates as for scenario-B strategy-1 ends at 150 kbps with PSNR value of 18.3 dB. And strategy-2 ends at 185 kbps with same PSNR value.

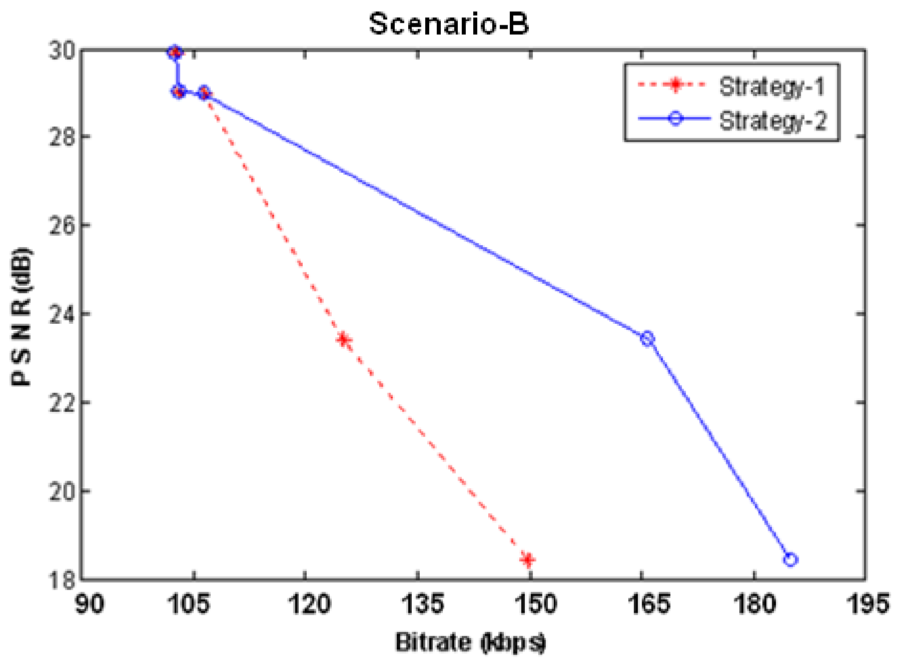


**Figure 6.8:** PSNR vs. Bitrate for Strategy-1 and 2 (Scenario-B)

- **Saving ratio obtained at various bitrates:**

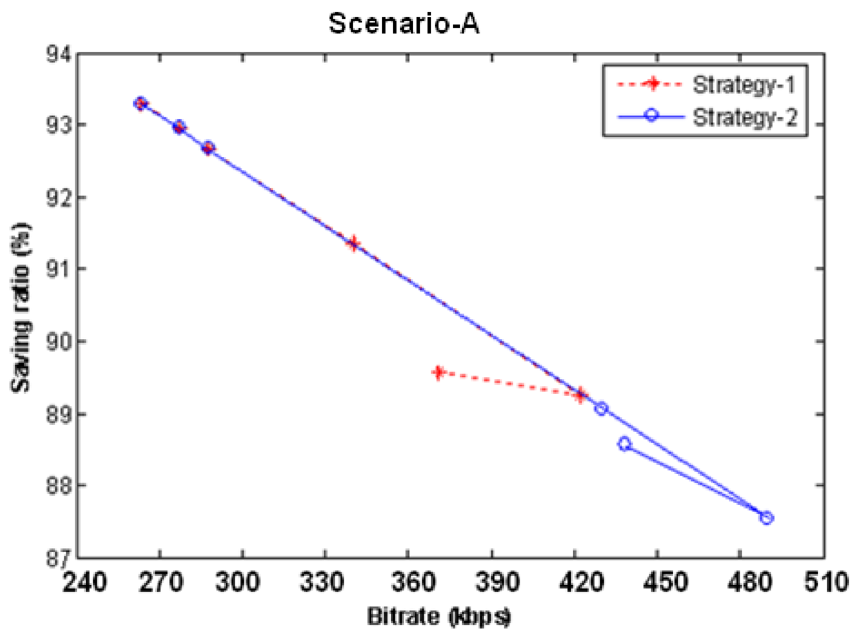


**Figure 6.9:** Saving ratio vs. Bitrates strategies comparison for Scenario-A

From Analysis of the saving ratio at various bitrates, it is clear that both the scenarios show the same relationship. A prominent decline can be seen in saving ratio with an increase in the bitrates (Figures 6.9 and 6.10). For scenario-A, saving ratio is decreased from 93.5% to 89.5% as the bitrate increases from 265 to 360 kbps for strategy-1. And for stargey-2 this decline is from 93.5% to 88% as the bitrate increases from 265 to 435 kbps. Like wise, for scenario-B there is a fall in saving ratio with the increase in bitrates for both the strategies but comparatively at lower bitrates, as shown in Figure 6.10.

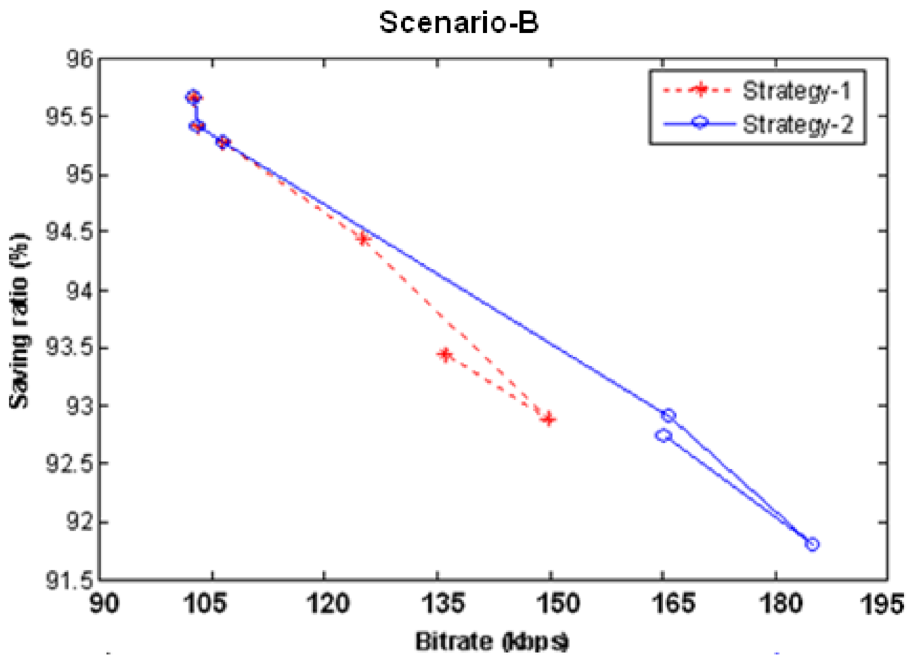


**Figure 6.10:** Saving ratio vs. Bitrates strategies comparison for Scenario-B

➢ **Distortion vs. Bitrates**

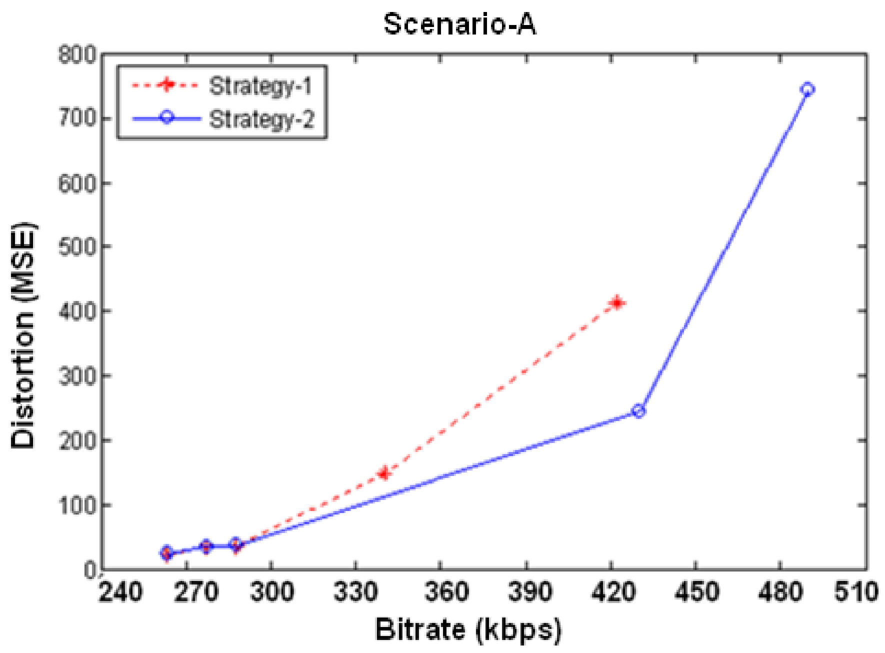


**Figure 6.11:** Distortion vs. Bitrates strategies comparison for Scenario-A

Analyzing the distortion achieved at various bitrates, Figure 6.11shows that more gain(less distortion) is obtained at lower bit rates while video frames start distorting as the bitrate increases. Figure 6.12 shows the same relationship between distortion and bitrate for scenario-B. According to Figure 6.12 both the strategies have less distortion at lower bitrates, but with an increase in bitrates distortion gradually increases and reaches its maximum value near 800.

The only difference between the two strategies is that this increase in distortion reaches early for strategy-1 as near bitrate of 145kbps while for strategy-2 it is on 190 kbps.

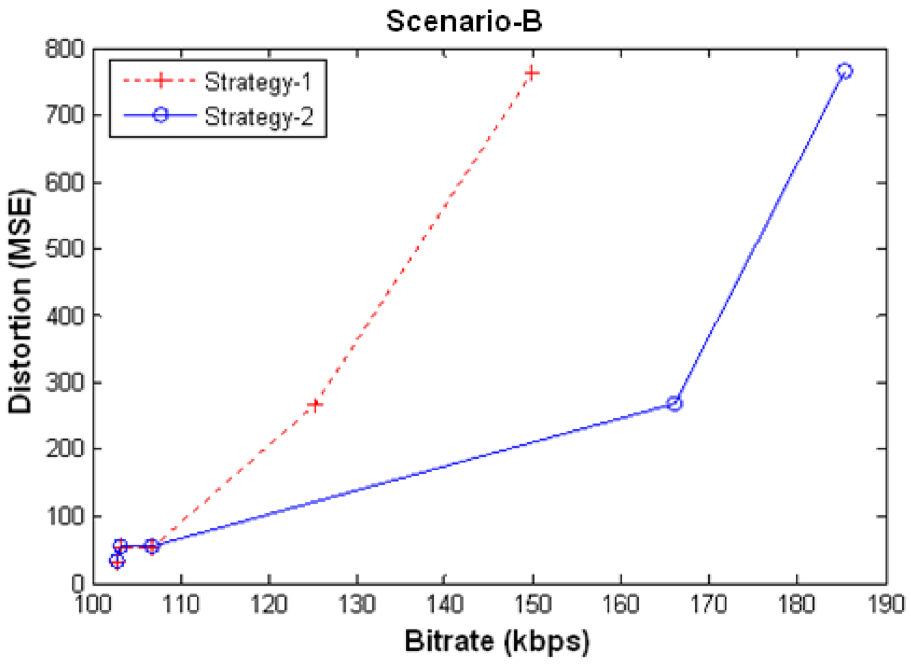


**Figure 6.12:** Distortion vs. Bitrates strategies comparison for Scenario-B

## 6.3 Comparison with other techniques

Comparing this MPEG approach with other approaches yields the following results as:

### 6.3.1 Comparison with Epipolar approach

One approach that has been discussed in literature survey chapter is implementation of epipolar geometry in mutli-terminal scenario. Comparing my MPEG technique with this technique shows that for both the techniques saving ratio starts decreasing with an increase in the bitrates. Results of the epipolar approach were formulated at various angles of the camera B from camera A, while my research work is compared on the basis of communication strategies with their respective working modes.

**Table 6.8: Results Comparison of Epipolar Approach and Proposed MPEG Approach**

| **Compression Parameters** | **MPEG Approach** | | | | **Epipolar Approach** | | | |
|---|---|---|---|---|---|---|---|---|
| Saving ratio vs. Bitrate | A-st1 | A-st2 | B-st1 | B-st2 | Angle=5 | Angle=15 | Angle=25 | Angle =35 |
| Saving ratio-start (%) | 93.2 | 93.2 | 95.7 | 95.7 | 69 | 65 | 60 | 57 |
| Saving ratio-end (%) | 89.5 | 88.3 | 93.4 | 92.7 | 22 | 21 | 20 | 18 |
| Bitrate-start (kbps) | 267 | 267 | 105 | 105 | 100 | 90 | 85 | 80 |
| Bitrate-end (kbps) | 366 | 429 | 135 | 165 | 620 | 620 | 620 | 610 |

## 6.3.2 Comparison with Model-based approach

Another approach that has been discussed in literature survey chapter is model-based approach in mutli-terminal scenario. Comparing the results of both techniques for P-frames shows the following trend.

**Table 6.9: Results Comparison of Model-based Approach and Proposed MPEG Approach**

| **Compression Parameters** | **MPEG Approach** | | | **Model-based Approach** | | |
|---|---|---|---|---|---|---|
| **PSNR Values(dB)** | **A-st1** | **B-st1** | **B-st2** | **BR=0.24** | **BR=0.32** | **BR=0.40** |
| **PSNR1** | **34** | **30** | **31** | **12.5** | **12.6** | **12.6** |
| **PSNR2** | **40** | **37** | **30** | **12.9** | **12.8** | **12.8** |
| **PSNR3** | **20** | **20** | **17.6** | **11.8** | **12** | **12** |
| **PSNR4** | **23** | **22** | **24** | **11.9** | **11.6** | **11.7** |

## 6.4 Summary

In this chapter, system performance has been analyzed and evaluated from different aspects of its functioning. Relationship of different parameters has also been shown graphically. Conclusively, it is stated that system gives good performance at lower bit rates for both the strategies under both working modes. Moreover, some system working setups give more gain and good PSNR as compared to other modes but all shows optimum results at lower bitrates.

Chapter 7

# Conclusion and Future Work

## 7.1 Discussion

Wireless sensor networks are multi camera setups containing power and energy constrained camera nodes. Each node carries the significant part of information which plays vital role in final output formulation. A single link is available to carry this huge load coming out of each and every camera sensor node. For a wireless sensor network where terminals are equipped with cameras and capturing the multimedia data, carrying this resultant load in available link capacity becomes a very serious problem.

Multimedia information is huge form of visual as well as audio and textual data. Additionally, this class of data comprises repetitive patterns and ambiguities. Capturing the video sequences from different terminals which are placed in space with some particular angle from the reference object results in some extra weight age of redundancy. Since each camera node is capturing the same scene from some different angle, there is a great degree of inter-node statistical redundancy among camera nodes findings. Likewise in an individual video sequence this repetition of patterns gives rise to another form of redundancy called temporal redundancy. And then in a single video frame, pixel values are highly correlated that result in spatial redundancy.

Usually these camera sensor nodes are deployed with video compression standards which tend to remove spatial and temporal redundancies that exist in each node video sequence. But when all nodes transmit their findings to a common link, there is still a great degree of repetition in the final data load due to inter-node statistical redundancy.

So for a wireless camera sensor network there has always been an emerging need for some standard that exploits these three kinds of redundancies. Beside this it must also provide good quality reconstructed video, as well as keeping the saving ratio at maximum. And finally, this all should be achieved keeping the communication among the camera sensor nodes at minimum.

## 7.2 Contribution of Project

Keeping all these requirements of a wireless sensor network in mind, an algorithm has been designed which is an extension of MPEG, a video compression standard for multi-terminal scenario. Camera nodes are allowed to communicate in two ways called communication strategy-1 and strategy-2. And in each scenario system is capable of working in two different modes or scenarios. Each working mode has its benefits and can be deployed in different environments with different set of requirements. Results have been computed for each working mode for both strategies. Both tabular as well as graphical results show that system performance highest gains are obtained at lower bit rates and there is a prominent distortion in video quality with increase in bitrates.

## 7.3 Future Directions

Since two communication strategies have been proposed in this research work. Overall out of the two communication strategies, strategy-1 should be adopted to get correspondence among video camera frames keeping the communication among the sensor nodes at minimum. Stargey-1 with working mode-1 gives good results.

So this combination should be exploited in future with more advanced and sophisticated compression techniques. Additionally, one drawback in working mode-1, that it takes too much processing time, should also be removed.

# References

[1] A. Sehgal, A. Jagmohan, and N. Ahuja, “Wyner-Ziv Cod-ing of Video: An Error-Resilient Compression Frame-work,”*IEEE Transactions on Multimedia*, vol. 6, no 2, pp. 249–258, April 2004.

[2] B. Girod, A. Margot, S. Rane, and D. Rebollo-Monedero, “Distributed video coding,” *Proceedings of the IEEE*, vol.93, no 1, pp. 71–83, January 2005.

[3] R. Puri and K. Ramchandran, “PRISM: A video coding architecture based on distributed compression principles,”submitted to *IEEE Transactions on Image Processing*.

[4] N. Gehrig and P. L. Dragotti, “DIFFERENT: Distributed and Fully Flexible image EncodeRs for camEra sensor NeTworks,” in *ICIP 2005*.

[5] Y. Yang, V. Stankovic, W. Zhao, and Z. Xiong, “Multiter-minal video coding,” in *Information Theory and Applica-tions Workshop*, San Diego, CA, January 2007.

[6] B. Song, A. Roy-Chowdhury, and E.Tuncel. “Towards A Multi-Terminal Video Compression Algorithm By Integrating Distributed Source Coding With Geometrical Constraints,” *in Journal of Multimedia, vol. 2, no 3, June 2007.*

[7] “The Epipolar Geometry Toolbox” by Gian Luca Mariottini and Domenico Prattichizzo, 1070-9932, 2005.

[8] B. Song, A. Roy-Chowdhury, and E.Tuncel. “Towards a multi-terminal Video Compression algorithm using epipolar geometry,” *IEEE Intl. Conf. On Acoustics, Speech and Signal Processing, 2006.*

[9] B. Song, A. Roy-Chowdhury, and E.Tuncel. “A multi-terminal model-based video compression algorithm,” *IEEE Intl. Conf. on Image Processing, 2006.*

[10] M. Flierl and B. Girod, “Coding of multi-view image sequences with video sensors,” in *IEEE Intl. Conf. On Image Processing*, *2006.*

[11] Image and Video Compression for Multimedia Engineering: Fundamentals, Algorithms and Standards by Yun Q.Shi and Huifang Sun

[12] R. Wagner, R. Nowak, R. Baranuik, “Distributed Image Compression for sensor networks using correspondence analysis and super resolution,” *in ICIP 2003*.

[13] T. Burger, “Multi-terminal source encoding,” in *the Information Theory Approach to Communications*, G. Longo, Ed. CISM Courses and Lectures 229.Springer, New York, 1978.

[14] Y. Oohama, “Gaussian multi-terminal source coding,” *IEEE Transactions on Information Theory,* vol. 43, no.6, pp. 1912-1923, November 1997.

[15] D. Slepian and J. K. Wolf, “Noiseless Coding of Correlated information sources,” *IEEE Transactions on Information Theory*, vol. 19. No 4. Pp 471-480, July 1973.

[16] N. Gehrig and P. L. Dragotti, “Distributed Sampling and Compression of Scenes with Finite Rate of Innovation in Camera Sensor Networks,” in *Data Compression Conference 2006*.

[17] X. Zhu, A.Aaron, and B. Girod, “Distributed Compression for large camera arrays” in *IEEE Workshop on Statistical Signal Processing, September 2003*.

[18] J. C. Dagher, M. W. Marcellin and M. A. Neifeld, “A Method for Coordinating the Distributed Transmission of Imagery,” in *IEEE Transactions on Image Processing*, pp.1705-1717, July 2006.

[19] http://www.mathworks.com/matlabcentral/fileexchange/13020

[20]http://www.mathworks.com/matlabcentral/fileexchange/8761-block-matching-algorithms-for-motion-estimation

[21] http://en.wikipedia.org/wiki/Moving_Picture_Experts_Group

# APPENDIX – A

## 1.Work Mode-1 Code

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%GOP Generator
for i=1:+6:215
[I1,p2,p3,p4,p5,p6,p21,p22,p23,p24,p25,p26]=GOP(i,m_name1,m_name2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%Working with Movie-1

[bufferI1,I_STREAM,idict]=IFrmProc(I1,array);
        XX(1)=length(I_STREAM);
        BB(1)=length(idict);
        bufferI1=double(bufferI1);
         I1d=IFrmDeco(I_STREAM,idict,array);

[bufferP2,motionVect,pstream,pdict]=encod_p(bufferI1,p2,mbsize,p,array);
        XX(2)=length(pstream);
        BB(2)=length(pdict);
        EE(2)=length(motionVect);
        P2d=PFrmDeco(pstream,pdict,I1d,array,motionVect,mbsize);

[bufferP3,motionVect,pstream,pdict]=encod_p(bufferP2,p3,mbsize,p,array);
        XX(3)=length(pstream);
        BB(3)=length(pdict);
        EE(3)=length(motionVect);
        P3d=PFrmDeco(pstream,pdict,P2d,array,motionVect,mbsize); %

[bufferP4,motionVect,pstream,pdict]=encod_p(bufferP3,p4,mbsize,p,array);
        XX(4)=length(pstream);
        BB(4)=length(pdict);
        EE(4)=length(motionVect);
        P4d=PFrmDeco(pstream,pdict,P3d,array,motionVect,mbsize);

[bufferP5,motionVect,pstream,pdict]=encod_p(bufferP4,p5,mbsize,p,array);
        XX(5)=length(pstream);
        BB(5)=length(pdict);
        EE(5)=length(motionVect);
        P5d=PFrmDeco(pstream,pdict,P4d,array,motionVect,mbsize); %

[bufferP6,motionVect,pstream,pdict]=encod_p(bufferP5,p6,mbsize,p,array);
        XX(6)=length(pstream);
        BB(6)=length(pdict);
        EE(6)=length(motionVect);
```

```
          P6d=PFrmDeco(pstream,pdict,P5d,array,motionVect,mbsize);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%Working with Movie-2

[bufferP21,I_STREAM,idict]=IFrmProc(p21,array);
           XX1(1)=length(I_STREAM);
           BB1(1)=length(idict);
           bufferP21=double(bufferP21);
          P21d=IFrmDeco(I_STREAM,idict,array);

[bufferP22,motionVect,pstream,pdict]=encod_p(bufferP21,p22,mbsize,p,array);
          XX1(2)=length(pstream);
          BB1(2)=length(pdict);
          EE1(2)=length(motionVect);
          P22d=PFrmDeco(pstream,pdict,P21d,array,motionVect,mbsize); %

[bufferP23,motionVect,pstream,pdict]=encod_p(bufferP22,p23,mbsize,p,array);
          XX1(3)=length(pstream);
          BB1(3)=length(pdict);
          EE1(3)=length(motionVect);
          P23d=PFrmDeco(pstream,pdict,P22d,array,motionVect,mbsize);

[bufferP24,motionVect,pstream,pdict]=encod_p(bufferP23,p24,mbsize,p,array);
          XX1(4)=length(pstream);
          BB1(4)=length(pdict);
          EE1(4)=length(motionVect);
          P24d=PFrmDeco(pstream,pdict,P23d,array,motionVect,mbsize); %

[bufferP25,motionVect,pstream,pdict]=encod_p(bufferP24,p25,mbsize,p,array);
          XX1(5)=length(pstream);
          BB1(5)=length(pdict);
          EE1(5)=length(motionVect);
          P25d=PFrmDeco(pstream,pdict,P24d,array,motionVect,mbsize);

[bufferP26,motionVect,pstream,pdict]=encod_p(bufferP25,p26,mbsize,p,array);
          XX1(6)=length(pstream);
          BB1(6)=length(pdict);
          EE1(6)=length(motionVect);
          P26d=PFrmDeco(pstream,pdict,P25d,array,motionVect,mbsize);
end
```

## 4 Work Mode-2 Code

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%Preparing First I-buffer

      framedata=aviread(m_name2,1);
      I1=frame2im(framedata);
      I1=imresize(I1,[128 128]);
```

```
I1=double(I1);
I1=I1(:,:,1);
[bufferI1,X,B,E]=encod_proc(I1,array_inter);
bufferI1=double(bufferI1);
XX(1)=length(X);
BB(1)=length(B);
EE(1)=length(E);
id=1;
decoder1(X,B,E,motionVect,id,array_inter);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%GOP Generator

for i=2:+6:36
[p2,p3,p4,p5,p6,I7,p22,p23,p24,p25,p26,I27]=GOP(i,m_name1,m_name2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%Working with Movie-1

[bufferP2,motionVect,X,B,E]=PFrmProc(bufferI1,p2,mbsize,p,array_inter);
XX(2)=length(X);
BB(2)=length(B);
EE(2)=length(E);
id=2;
decoder1(X,B,E,motionVect,id,array_inter);

[bufferP3,motionVect,X,B,E]=PFrmProc(bufferP2,p3,mbsize,p,array_inter);
XX(3)=length(X);
BB(3)=length(B);
EE(3)=length(E);
id=3;
decoder1(X,B,E,motionVect,id,array_inter);

[bufferP4,motionVect,X,B,E]=PFrmProc(bufferP3,p4,mbsize,p,array_inter);
XX(4)=length(X);
BB(4)=length(B);
EE(4)=length(E);
id=4;
decoder1(X,B,E,motionVect,id,array_inter);

[bufferP5,motionVect,X,B,E]=PFrmProc(bufferP4,p5,mbsize,p,array_inter);
XX(5)=length(X);
BB(5)=length(B);
EE(5)=length(E);
id=5;
decoder1(X,B,E,motionVect,id,array_inter);

[bufferP6,motionVect,X,B,E]=PFrmProc(bufferP5,p6,mbsize,p,array_inter);
XX(6)=length(X);
BB(6)=length(B);
```

```
EE(6)=length(E);
id=6;
decoder1(X,B,E,motionVect,id,array_inter);

[bufferI7,X,B,E]=encod_proc(I7,array_inter);
bufferI7=double(bufferI7);
XX(7)=length(X);
BB(7)=length(B);
EE(7)=length(E);
id=7;
decoder1(X,B,E,motionVect,id,array_inter);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%Working with Movie-2

[bufferI2,motionVect,X,B,E]=PFrmProc(bufferI1,I2,mbsize,p,array_inter);
XX1(1)=length(X);
BB1(1)=length(B);
EE1(1)=length(E);
id=8;
decoder1(X,B,E,motionVect,id,array_inter);

[bufferP22,motionVect,X,B,E]=PFrmProc(bufferI2,p22,mbsize,p,array_inter);
XX1(2)=length(X);
BB1(2)=length(B);
EE1(2)=length(E);
id=9;
decoder1(X,B,E,motionVect,id,array_inter);

[bufferP23,motionVect,X,B,E]=PFrmProc(bufferP22,p23,mbsize,p,array_inter);
XX1(3)=length(X);
BB1(3)=length(B);
EE1(3)=length(E);
id=10;
decoder1(X,B,E,motionVect,id,array_inter);

[bufferP24,motionVect,X,B,E]=PFrmProc(bufferP23,p24,mbsize,p,array_inter);
XX1(4)=length(X);
BB1(4)=length(B);
EE1(4)=length(E);
id=11;
decoder1(X,B,E,motionVect,id,array_inter);

[bufferP25,motionVect,X,B,E]=PFrmProc(bufferP24,p25,mbsize,p,array_inter);
XX1(5)=length(X);
BB1(5)=length(B);
EE1(5)=length(E);
id=12;
decoder1(X,B,E,motionVect,id,array_inter);

[bufferP26,motionVect,X,B,E]=PFrmProc(bufferP25,p26,mbsize,p,array_inter);
```

```
XX1(6)=length(X);
BB1(6)=length(B);
EE1(6)=length(E);
id=13;
decoder1(X,B,E,motionVect,id,array_inter);

[bufferI27,X,B,E]=encod_proc(I27,array_inter);
bufferI27=double(bufferI27);
XX1(7)=length(X);
BB1(7)=length(B);
EE1(7)=length(E);
id=14;
decoder1(X,B,E,motionVect,id,array_inter);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            I1=I7;
            bufferI1=bufferI7;
            I1d=I7d;
            I2=I27;
           bufferI2=bufferI27;
           I2d=I27d;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
end
```