A Spatial Scalable Video Coding with Selective Data Transmission using Wavelet Decomposition

by

Lakshmi Veerapandian

Bachelor of Engineering (Information Technology)
University of Madras, India. 2004.

Thesis submitted for the degree
of

Master of Engineering Science
in
School of Electrical and Electronic Engineering

The University of Adelaide

March 2010

1. Introduction

This chapter gives a short synopsis of the research. It starts with the motivations behind this research work and the gap statement, followed by the discussion of prevailing solutions and the limitations of these approaches. This is followed by the overview of the research, and lastly the outline of the thesis.

1.1 Motivation and Problem Statement

In recent years, the popularity of the Internet has seen a tremendous growth and along with it the use of multimedia rich applications, like video conferencing, distance learning, news, entertainment, training and on-demand video streaming, has also increased exponentially. In particular audio-visual applications have been regarded highly for instant access to multimedia information. Users like to access video material, irrespective of their format, from any type of output device while on the move. This has been the motivation of this research: to propose an adaptive form of video that could be accessed anytime anywhere in any type of format, thus providing the user with greater flexibility and mobility during information access.

The growth of the Internet has infused subsequent development in networks and user devices to support diversity, which requires transmitting of video over various networks to different user devices with different user specifications, as shown in Figure 1.1. In the figure, a scenario of compressed video transmission from the server through Internet, which has varying networks, to the different receiving devices is represented. This is an example to show the diverse nature involved in video transmission. The adaptation of video to such diversity is imposed by two main factors: heterogeneity of networks and diverse user device capabilities.

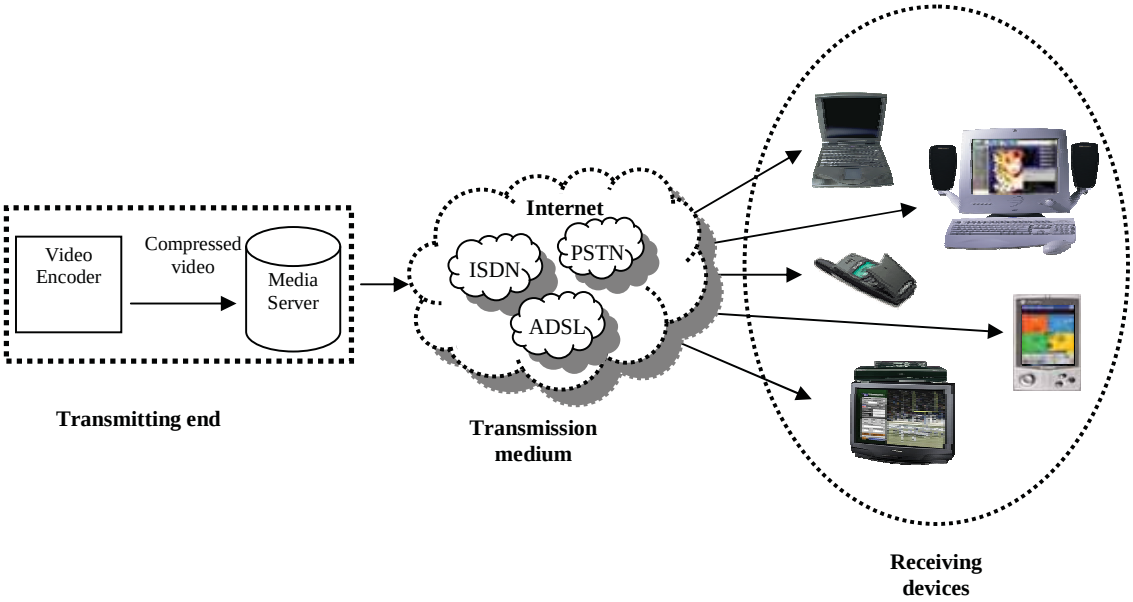


Figure 1.1 Heterogeneous nature of video delivery

1.1.1 Network Constraints

Bandwidth Gap. There are different types of networks available today with different transmission capacities. The bandwidth, a measure of the transmission capacity of the network, ranges from as low as 54 kbit/s to as high as 10 Mbit/s or more. The higher the data rate, the higher is the potential quality of the video as the data rate is directly proportional to the video compression, quality and resolution. In a dial-up connection, the bandwidth being very limited in terms of 56 kbit/s, the user cannot receive a video at a rate of 300 kbit/s and has to settle for video with lower quality and resolution. On the other hand, with a higher connection speed, such as ADSL or T1 connections, the user can enjoy video with higher quality, frame rate and higher resolution. Maintaining a balance between video bit rate and bandwidth is a difficult task.

Unpredictable nature of the Internet. The Internet, being a packet network, cannot guarantee quality of service for data transmission in general, and in particular stable data rates for on-demand video streaming. Small loss of data packets or delay can cause significant degradation in the quality of the video. During periods of network congestion the data rate can vary wildly and packets can be lost, resulting in imperfect transmission,

potentially to the point of complete failure of useful transmission. Therefore, video transmission must be tailored to handle any such unpredictable situation of the Internet.

1.1.2 User Constraints

End users may access information from different types of devices such as Personal Digital Assistant (PDAs), Personal Computers (PCs), High Definition TV (HDTV) or mobile phones. These multimedia-rich devices differ in display resolution, power, memory and processing capacity. For example, the screen sizes for PCs are in the range of 800x600 to 1920x1080, for Handheld PCs in the range 480x240 to 640x240, for PDAs in the range of 160x160 to 320x240, and cellular phones from 128x128 to 480x640 (Mohan *et al.* 1999, GSM Arena 2000). These differences cause a restriction in the information access and demand a device specific format of data. For example, people like to use handheld devices like cellular phones and PDAs for communications such as video conferencing, video chatting, on-demand video streaming, and live video podcasts of news, sports and television programs. Most of these devices have limitations in processing capacity, power and display resolution which places a restriction on the amount of data received and processed by these devices. Even though the bandwidth of the network is high, the device has low processing capability and screen sizes cannot process video of higher quality and bit rate without resizing their resolution and bit rate. Likewise changing to different devices while accessing the same information tends to have a similar impact. Even with high network bandwidth and high-end devices the user might just want to have a lower quality version of the video, as the user might have different priorities. Thus the video delivered to the user must be compatible with the limitations of the different devices and user preferences.

1.2 Existing Solutions and its Limitations

Commonly practiced solutions are storing multiple copies of the same video on an image server, and transcoding. However each has its own drawbacks. Another alternative lies in having a scalable representation of video with the ability to adapt to varying network and user requirements. The scalable video representation has greater advantage than transcoding for its reduced storage, flexible adaptation and advanced feature sets.

Multiple entries. The traditional approach was to code the video to a specific format to meet each constraint. This approach requires huge computational and storage space and it is not a flexible solution.

Transcoding. Transcoding is one of the conventional ways of making video compatible with both network and user constraints. It is the process of converting the compressed video stream into the required video specification. A transcoder unit is required for any conversion. For example, when video is delivered to different user terminals, a transcoder is required for each and every terminal, implying that video is subjected to transcoding every time a change occurs. Even though this process addresses the problem, it is not an efficient solution, as it introduces delay and drift problems. The major issue is that the transcoding process requires repetitive decoding and encoding of video whenever scaling is required.

Scalable approach. The above-mentioned techniques help to provide an adaptive solution but lack flexibility and efficiency. In contrast to these approaches, scalable decimation has emerged as a flexible and effective solution. In a scalable approach video can be coded once and can be decoded multiple times from the same source without the need of transcoding every time. Scalability can be broadly classified into three types: spatial scalability (for change in resolution), temporal scalability (for change in frame rate) and SNR scalability (for variation in data rate or quality).

Recent standards offer scalability to a certain extent, like the fine granular scalability (FGS) in the MPEG-4 standard. However, in FGS, scalability increases the complexity of the encoding algorithm, thereby resulting in quality degradation (Li 2001). Also other scalable video coding techniques based on the layered representation (Woods 1991, Uz *et al.* 1991, Taubman and Zakhor 1994) tend to have a drift problem.

A more practical solution to scalability problem can be obtained by using a wavelet-based approach, which gives superior performance in video as in images (Shapiro 1993,

Taubman 2000). Compared to the conventional discrete cosine transform (DCT) the wavelet transform provides better flexibility. In addition, the wavelet transform can also be used for multiresolution analysis (Mallat 1989a), which is a key scalability feature. The multiresolution approach combined with the wavelet transform (Zhang and Zafar 1992) has drawn much research interest in scalable video coding techniques including the MPEG standard (Chen and Woods 2002). In still image coding there has emerged a new standard called the JPEG2000 (Taubman and Marcellin 2002), which provides flexible adaptation due to its excellent scalability features. It uses discrete wavelet transform (DWT) and multi-scale representation. Motion JPEG2000, an extension of JPEG2000 in video, has been proven to be as efficient as H.263 in intramode (Marpe *et al.* 2004). Unfortunately Motion JPEG2000 does not remove temporal redundancy.

Although there have been many advances in providing an efficient scalable solution for video adaptation in recent years, discussed in Chapter 4, all these scalable algorithms lack an efficient motion model or a scalable motion vector representation, which is primordial for combating temporal redundancy, a key aspect of video coding.

1.3 Research Objective

In this thesis, we present a framework for a scalable video coding which can adapt to both network constraints and user requirements. In addition, the video data can be transmitted in different levels of quality and resolution depending upon the available bandwidth of the network. This adaptive nature helps to delivery video in an acceptable form during varying network rate or during network congestion.

The first part of this research focuses on providing scalable video that can adapt to varying user terminals, known as resolution scalability. This is achieved by the use of wavelets, which provides a flexible scalable approach. In providing multi-dimensional representation in video another factor has to be taken into account, which is the correlation of the video frames along the time line. Multiresolution motion estimation is used to effectively utilize the correlation of video frames thereby reducing the temporal redundancy. This helps to provide an efficient coding solution for resolution scaling.

The second part focuses on providing selective transmission of video based on three factors: quality, resolution and bandwidth of the network. In order to achieve this we have used two thresholds to control these factors. Upon varying the thresholds, different combinations of video can be obtained in terms of quality and resolution. For example, for high bandwidth and resolution scenarios the threshold can be moved to the maximum thereby allowing the video to have higher resolution and higher quality to fit the available bandwidth.

The application of multiresolution motion estimation causes translation error across different resolution levels. The translation errors are rectified using two different approaches: (1) motion vector replacement and (2) block replacement. One of the two thresholds is used for controlling the error correction factor. The factor determines what percentage of error in the given frame can be corrected. As error correction slightly increases the amount of data to be transmitted, the error correction factor is directly proportional to the available bandwidth. Thus the higher the bandwidth, the more blocks are updated and the more error robust the video becomes. We have also presented the comparison between the two different error correction methods, discussing the benefits and the drawbacks of these two methods.

1.4 Outline of the Thesis

The remainder of the thesis is organised as follows:

- Chapter 2 gives background information of video, network and video standards.
- Chapter 3 gives an insight into the concepts and techniques involved in the research for the better understanding of the thesis, like video coding, motion estimation, and motion compensation.
- Chapter 4 presents existing techniques that are relevant and similar to this research work. Their approaches, application, advantages and drawbacks are also debated here.

- Chapter 5 gives a detail description of the spatial scalable video coding framework and the different algorithms simulated prior to the final approach and its interpretations.
- Chapter 6 presents the simulation results, its interpretations and its evaluations.
- Chapter 7 summarizes the outcome of the research and its application and possible development of this research in the future.

2. Background

This chapter gives an overview of the fundamental concepts of video, video coding and video standards.

2.1 Fundamentals of Video

Video is a collection of still-images played in a timed-sequence. Playing a series of images in a sequence replicates motion. The human eye has a property of capturing motion at a slower pace, so when continuous set of images are displayed at a faster rate it tricks the Human Visual System (HVS) into interpreting them as motion. Hence continuous images have to be displayed at a rate greater than 20 frames per second (fps) for the human eye to perceive the video as a smooth motion. The rate at which images are displayed determines the frame rate of the video.

Video exists in two forms: analog and digital. Analog form was used traditionally for video recording, storage and transmission. Analog video (Ghanbari 1999) is a continuous one-dimensional electrical signal representing the time-varying images as intensities over time. In analog video the contents of each frame (or image) are displayed, on the screen, in the form of horizontal lines scanning from top-left hand corner to bottom-right in a zigzag manner. If the image is formed by a single continuous scanning then it is called *progressive scanning.* Otherwise if it is formed by two successive scanning interleaving each other it is called *interlaced scanning.* As humans perceive visual information in analog format, analog form was the best form of representation due to its simplicity and accurate representation, but it is difficult to manipulate, transmit, and distribute among different users and it is easily prone to interference and distortions. The three commonly used analogue colour video standards, *National Television System Committee* (NTSC), *Phase Alternation Line* (PAL), and *Sequential Couleur Avec Memoire* (SECAM) (Netravali and Haskell 1995) are used in different parts of the world and are mutually incompatible. While PAL and SECAM use the same resolution and frame rate, their colour coding schemes are different; NTSC has a different resolution and frame rate.

Digital video (Al-Mualla *et al.* 2002) is easier to encode, handle and compress as the signals are converted to digital form, allowing a universal medium of information

processing, transfer and storage. Noise interference is easily avoided in digital video, although distortion due to compression can occur. Digital video can be represented in different formats to suit different needs. The word ‘video’ in the rest of this thesis will refer to digital video.

Video as mentioned earlier is composed of still images called *frames*. The number of pixels in the horizontal and vertical axis represents the size of the frame. The frames are composed of *macro blocks*, which are of sizes 16x16 pixels or greater. The macro blocks are in turn composed of group of *microblocks* of size 8x8 pixels, where *pixels* are the smallest classification of a frame.

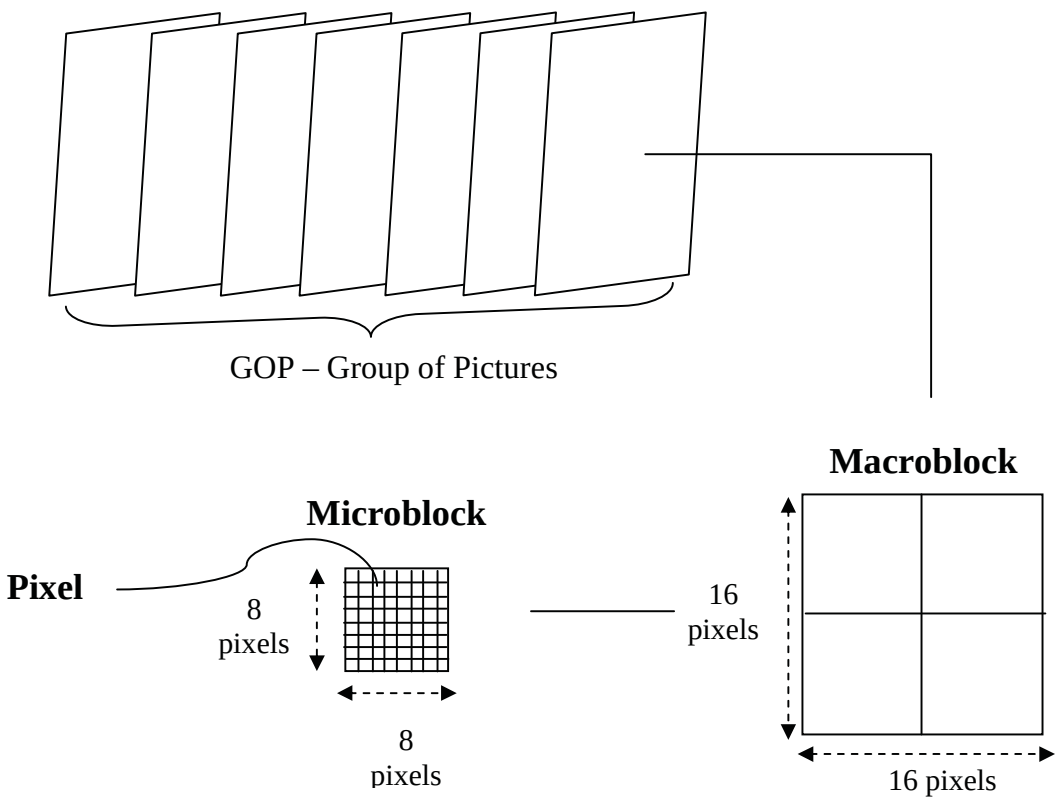


Figure 2.1: Building blocks of digital video

2.2 Characteristics of Video

Video is characterized by the following parameters: (1) colour representation, (2) resolution, (3) frame rate, (4) bit rate, and (5) quality. A variation in these parameters helps in creating video in different formats to suit different applications, users and network requirements.

Colour representations

Each pixel, in colour video, is made up of three colour components Red-Green-Blue (RGB), or luminance (Y) and two chrominance components (UV or CrCb). Luminance refers to the brightness of each pixel in the image (so called "black and white") and chrominance, the colour information of the image. The human eye is less sensitive to the colour detail than the details in the luminance plane; hence video coding operations like motion estimation are performed only on the luminance component of the image.

The colour depth of an image is determined by the number of bits used to represent a pixel. In the RGB colour model the pixels are made up of the combination of red, blue and green, requiring eight bits for each colour, i.e. 8 *bits per pixel per colour plane.* So with 8 bits we have 2^8 possible values for each of red, green and blue pixel value. In combination, this means 3x8=24 bits per pixel (bpp), representing over 16 million colour/luminance values. This is significantly more information than the Human Visual System can process, leaving room for compression techniques, which discard information without a visual impact for the end user.

Resolution

The size of the digital video is measured by the number of pixels in horizontal (width) and vertical (height) direction contained in its individual frames, generally referred as *video resolution*. For example, if the resolution of the video is 144x176 then it means that the video is made of frames with 144 rows of 176 pixels per line. There exist different video formats to suit different applications, networks and terminals. The most commonly used digital video formats (Ghanbari 1999) are

- CCIR-601 (1990) developed by the *International Consultative Committee for Radio* (CCIR) (currently known as *International Telecommunications Union – Radio Sector* (ITU-R)) was used mainly for broadcast-quality applications.
- *Source Input Format* (SIF) with lower resolution for storage applications.
- *Quarter-SIF* (QSIF) a lower version of SIF
- *Common Intermediate Format* (CIF) was developed for video conferencing application with its higher and lower versions such as 4CIF, *Quarter-CIF* (QCIF), *Sub-Quarter CIF* (SQCIF), the smallest standard image size for mobile network applications, and
- *High Definition Television* (HDTV) and its variations.

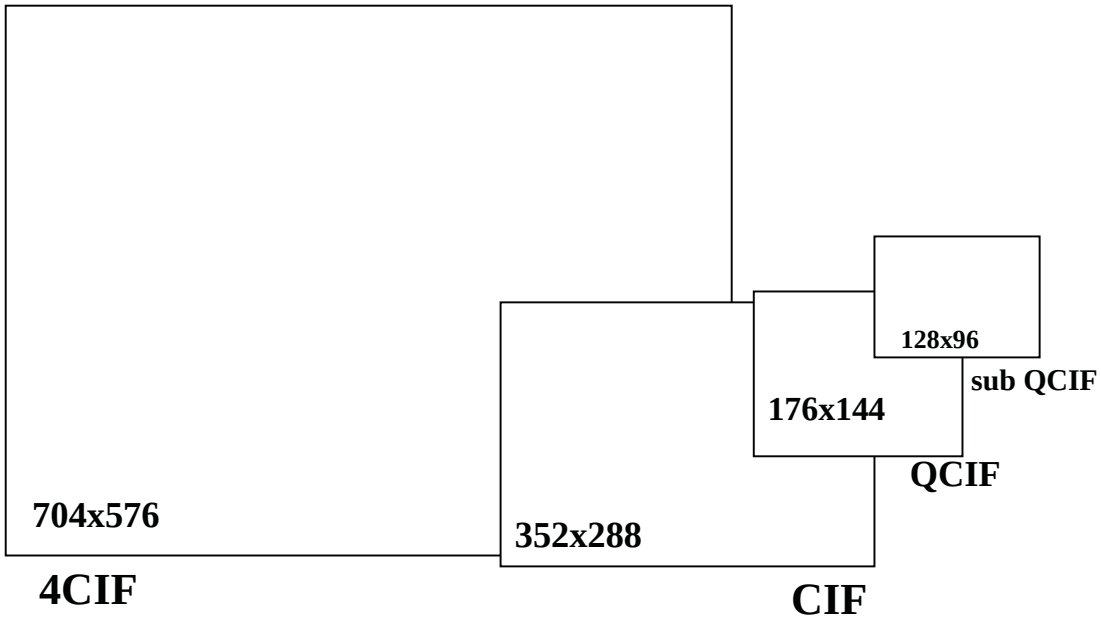


Figure 2.2: Resolution representation of CIF format and its family

Frame Rate

The number of frames displayed per second is known as *frame rate*, fps in short. This parameter is directly linked to the motion of the video. That is for slow motion the frame rate of the video should be low compared to a fast motion where the frame rate is high. Therefore for a smooth motion, the frame rate should in the order of 20-35 fps. Full motion video usually has a frame rate of 30 fps, while traditional movies have 24 fps

frame rate. The NTSC television standard uses 30 fps, while PAL and SECAM run at 25 fps. Very low bit-rate video, for example for handheld devices can be coded at 5-15 fps, with noticeable jerkiness in any motion.

Bit Rate

The rate at which video information (bits) is delivered is known as *bit rate*. It is usually measured in bits per second or bit/s. The number of bits transmitted depends on the size of the video, its frame rate, colour depth and the bits per pixel count. Therefore,

$$\text{Bit rate} = \text{Resolution} \times \text{bits per pixel} \times \text{Colour depth} \times \text{Frame rate (bit/s)} \quad (2.1)$$

For example, in the absence of any data compression, a video of size 320x288 pixels with 8 bits/pixels of RGB colour format which runs at 25 fps will have a bit rate of

$$\text{Bit rate} = (320\text{x}288) \times 8 \times 3 \times 25$$
$$\approx 55 \text{ Mbit/s}$$

Thus 55 Mbits per second are required to transmit a video of resolution 320x288 with 25 fps. As the resolution and frame rate increases, the bit rate increases.

Quality

The quality of the video refers to the visual clarity of the image. Quality incorporates not only the resolution and frame rate, but also consideration of the removal of redundant information, which has the least visual impact for the viewer. Such "lossy compression" includes consideration of reduced resolution of chrominance data, and coarse quantization of fine detail, which the eye tends not to notice. In general, there is a direct correlation between bit rate, resolution, frame rate and quality.

While the importance of broader image quality is acknowledged, in this thesis, only frame rate and resolution are considered as quality parameters, which is to say that lossy compression techniques are a secondary consideration. Hence we describe a CIF resolution video of resolution 352x288 pixels to be of lower quality than 4CIF (704x576).

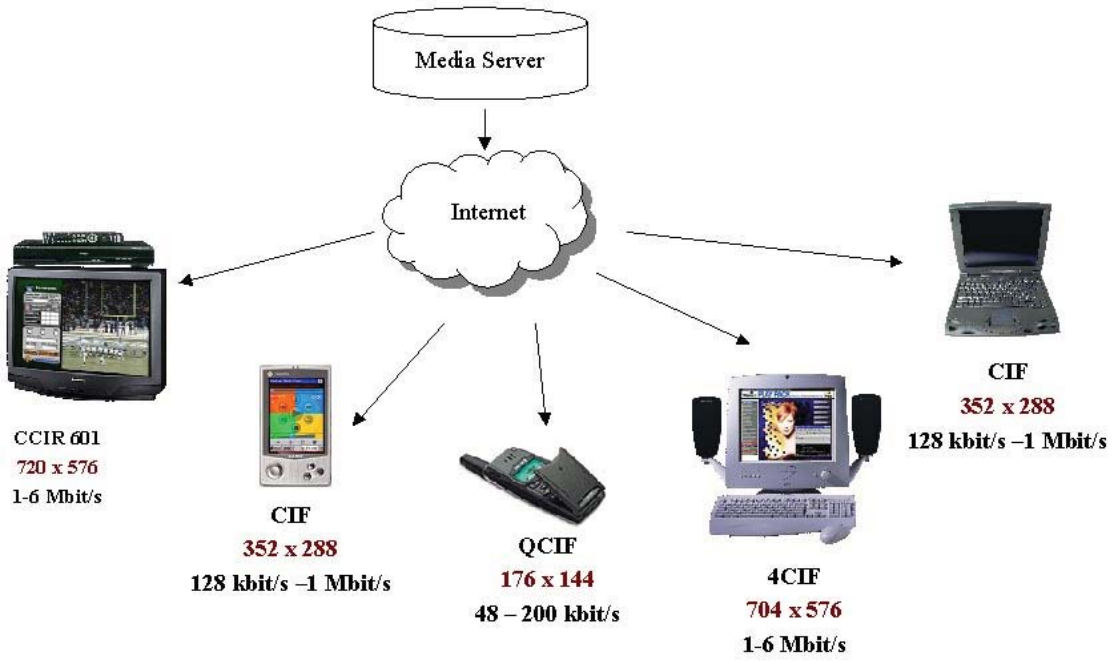


Figure 2.3: Different user device specifications

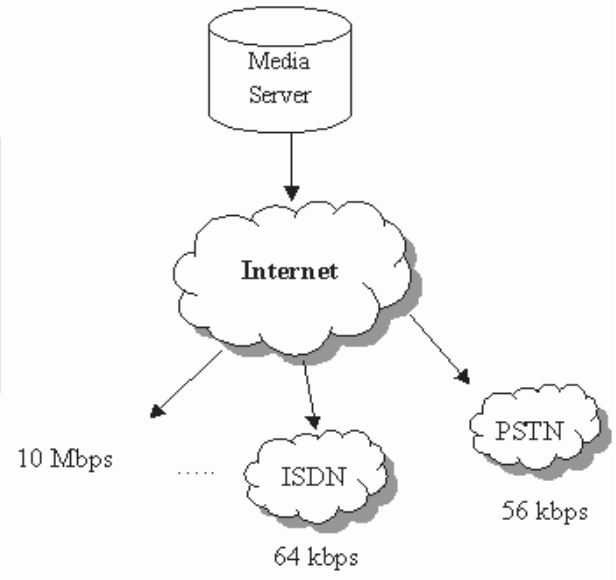
Figure 2.3 shows a range of different user devices with different resolutions and hence video qualities.

2.3 Network Variations

Telecommunication network links exists in many forms, as shown in Figure 2.4, to suit different user needs and application requirements. Data rates range from low speed dial-up connection to high-speed broadband connection such as ADSL[1], and ADSL 2+. Links can also be wired or wireless, and can be solely terrestrial or connected via satellites. The capacity of the network is determined by an important factor called *bandwidth*. It is the amount of data that can be transmitted through the medium at a time. Bandwidth is measured in bits per second. This data-transfer rate varies over time, as it degrades with an increase in traffic, due to sharing of resources. Hence end users' bandwidths are low during peak hours, when more users access the network.

[1] Asymmetric Digital Subscriber Line (ADSL)

NOTE:
This figure is included on page 14 of the print copy of the thesis held in the University of Adelaide Library.



Sourced from (Amon and Pandel 2003)

Figure 2.4: Transmission Capacity of the Network

2.4 Video Standards

Digital video standardization began in early 1980s initially by the *International Telegraph and Telephone Consultative Committee* (CCITT), which is currently know as *International Telecommunications Union – Telecommunication Standardisation Sector* (ITU-T), followed by CCIR, the *International Organisation for Standardisation* (ISO), and the *International Electrotechnical Commission* (IEC). They have introduced a number of international video coding standards such as: H.120, H.261, MPEG-1, MPEG-2, MPEG-4, H.263, H.263+, H.263++ and H.26L, of which the mostly widely used standards are outlined briefly here.

H.120

H.120 was the very first video coding standard developed in early 1980s by CCITT. It was mainly developed for video conferencing application with a typical bit rate of about 1.5 Mbit/s and 2 Mbit/s to suit aggregated telephone links.

H.261

This standard was designed by ITU-T (1990) for video telephony and video conferencing applications. It supports lower bit rate of around px384 kbit/s, with p between 1 and 5,

and px64 kbit/s, where p is between 1 and 30. Hence H.261 is also known as px64. It supports two types of resolution formats: CIF and QCIF.

H.263

H.263 (ITU-T 1996) was developed in 1996 for very low bit rate applications supporting bit rates lower than 64 kbit/s. It has combined features of H.261 and MPEG but with very low bit rates. In addition to CIF and QCIF formats, as supported by H.261, H.263 supports sub-QCIF, 4CIF and 16CIF formats.

H.264

H.264/AVC (Advanced Video Coding) was developed by the ITU-T and ISO/IEC MEPG organizations. H.264 presents a number of advances in video coding in terms of coding efficiency, error robustness and flexibility. It is based on block based motion compensation approach and hybrid video coding schemes, and is a harmonization of H.26L and MPEG-4 Very Low Bit-Rate Video (VLBV).

H.26L

H.26L, also known as H.263L, provides better quality and additional functionalities than the other existing standards. It has improved error robustness and adaptive rate control mechanism. Similar to H.261 and H.263, H.26L targets very low bit rate applications.

MPEG -1

MPEG-1 (ISO/IEC 11172) was created by the *Moving Picture Experts Group* (MPEG) under ISO. It was designed for storage applications with VHS-quality video and fixed data rate up to 1.5 Mbit/s, which could be played from CD-ROM. It supports 320x240 resolutions and a frame rate of 30 fps (similar to NTSC). It is much similar to H.261 but has a more advanced algorithm.

MPEG-2

MPEG-2 (ISO/IEC 13818, Haskell *et al.* 1997), a standard for high quality video compression was mainly designed for broadcast related applications. It has better video

quality compared to MPEG-1 and supports higher bit rate of 2-10 Mbit/s. MPEG-2 has greater advantage than MPEG-1 at higher bit rate but at lower bit rate it gives the same performance as MPEG-1. In other words, MPEG-2 provides variable bit rates. It supports spatial and temporal scalability especially for low resolution devices, and SNR[2] scalability. MPEG-2 is widely used in digital television and DVD.

MPEG-4

Introduced in 1999, MPEG-4 (ISO/IEC 14496) has similar properties as the other MPEG standards but provides video coding in a whole new perspective. It uses *object-based* or *content-based representation*, to represent the different object in the video. It supports a wide range of applications as it supports both higher bit rate and lower bit rates with quality images. MPEG-4 has better error resilience features. It supports a wide range of data rates ranging from 5 kbit/s to 4 Mbit/s. In particular, Very Low Bit-Rate Video (VLBV) is supported, related closely to (but incompatible with) H.26L.

JPEG2000

JPEG2000 (ISO/IEC 15444-1:2000, Taubman and Marcellin 2002) is actually a still-image format, but is worth mentioning here in the context of spatial scalability. The JPEG2000 was developed by the ISO and IEC. It is based on discrete wavelet transform (DWT), context modeling, arithmetic coding and post-compression rate allocation. JPEG2000 supports error resilience, region of interest, random access, multi component images and palletized colour, and many other feature sets.

Of particular interest is that JPEG2000 was conceived as a mechanism for decimated retrieval of an image from a complete data structure. Thus, instead of sending a complete image file, only the parameters required to meet the end user's requirements are transmitted. This might mean transmitting at a lower resolution, transmitting only a portion of the image, or transmitting only selected colour planes.

[2] SNR – signal-to-noise ratio, known as rate scalability

Motion JPEG2000

In video, the same concept of JPEG 2000 is delivered in the form of Motion JPEG 2000. Motion JPEG 2000 applies JPEG 2000 to individual frames and sends them across to the decoder, where the frames are played continuously. Motion JPEG 2000 does not apply motion estimation to the frames, which is the major drawback of this codec, and has poor compression efficiency.

3. Video Coding

This chapter provides some background in video coding and an overview of the different video coding systems and their mechanisms.

3.1 Basics of Video Coding

Video is a sequence of frame and involves huge amount of data. The size of the original video (uncompressed video) is very high in the order of several hundred Megabits per second, whereas the available bandwidth is very small, in the order of 10 kbit/s and 10 Mbit/s (Al-Mualla *et al.* 2002). It is impossible to transmit video in its original form across such low bandwidth networks, hence it has to be compressed to a smaller accessible rate. The main aim of a video coding system is to reduce the size or the bit rate of the video while maintaining acceptable quality and complexity. The data rate is reduced by removing unnecessary information such as spatial and temporal redundancies and information that is insignificant to the human visual system. A video coding system consists of an encoder unit at the transmitter end and a decoder unit at the receiver end. The encoder helps to code the original video into a compressed video stream for transmission across the network and the decoder decodes the compressed video stream back into its original form for display at the user end. A block diagram of an encoder is given below.

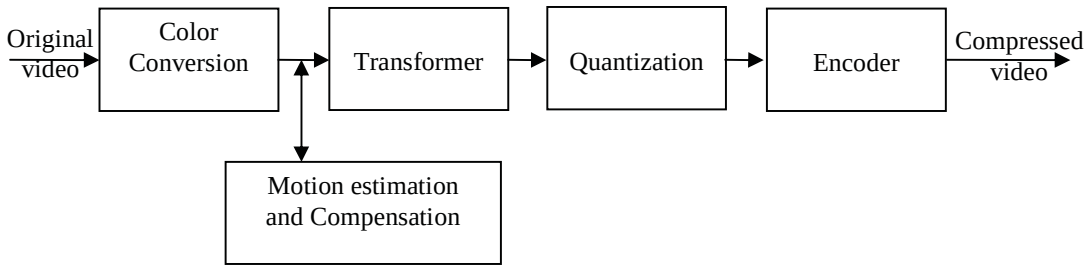


Figure 3.1: Video encoder architecture

The original video is first colour transformed from RGB to YUV or gray scale and then passes into the transformer unit. Here, using a spatial frequency transformation such as Fourier, Discrete Cosine or Wavelet transformation, the video is transformed into a different representation for the easy removal of redundant spatial information. The

redundancy between frames (temporal redundancy) is removed by performing motion estimation with correction of significant differences. The transformed coefficients are then passed through quantization in which unimportant visual information is removed. Finally, the encoder converts the series of symbol representing video data to a compressed bit stream format. The decoder unit has the similar process as in the encoder but in the reverse order.

3.2 Spatial Video Transformations

3.2.1 Transform Coding

Transform coding is the one of the traditional form of video coding, which has been used in most of the video coding standards. These techniques are usually based on orthogonal linear transforms, which uses block transformations. The earliest transform coding technique used was Karhunen-Loeve transform (KLT) and the other forms are Discrete Fourier Transform (DFT) and the Discrete Cosine Transform (DCT) (Ahmed *et al.* 1974). Among these the most widely used technique is the DCT transform, which is a special case of KLT (Jain 1989). As the DCT transform is a block based transform, it produces many disadvantages. It introduces a blurring effect due to the truncation of high-frequency components, and blocking artifacts, which appear around the edges of the blocks due to the independent processing of blocks (Al-Mualla *et al.* 2002). The DCT process is also computationally complex and supports little or no flexibility in terms of scalable option.

3.2.2 Subband Coding

In subband coding (Woods 1991), the images are sampled through a set of bandpass filters, which decompose the images into a set of bandlimited images called subbands. This technique was first introduced to image coding by Woods and O'Neil (1986). The main concept of subband coding is to treat the different subbands differently. The images are decimated and interpolated by downsampling and upsampling. The most ideal filter for subband coding is the Quadrature Mirror Filter (QMF). Subband coding also supports the property of multiresolution analysis if non-uniform subband decomposition is

performed. The discrete wavelet transform is a special case of non-uniform subband decomposition. As subband coding achieves energy compactness by filtering, it does not suffer from blocking artifacts as in transform coding.

3.2.3. Wavelet Transform

The Discrete Wavelet Transform (DWT) has emerged as a popular and efficient approach in image and video coding due to its superior decomposition properties, flexible signal representation and its ability to adapt to human visual characteristics (Antonini *et al.* 1992, Wang *et al.* 2002). DWT provides better compression rates and image quality than discrete cosine transform (DCT) (Xiong *et al.* 1999, Taubman and Marcellin 2002). The wavelet transform can be applied as 2D transform on individual images or on motion compensated images, or as a three dimensional signal transform (space and time) on a group of images.

3.2.4. Subband/Wavelet Encoders

The decomposed subband or wavelet images can be encoded using various methods. The most commonly used embedded encoding schemes are EZW, SPIHT, and EBCOT. The Embedded Zero-tree Wavelet (EZW) technique was first introduced by Shapiro (1993). The embedded bit stream was obtained by the combination of a zero-tree structure and bit plane coding in the wavelet domain. This fuelled further research in embedded coding which led to the development of Set Partitioning in Hierarchical Trees (SPIHT) (Said and Pearlman 1996) algorithm and, recently, the Embedded Coding with Optimized Truncation (EBCOT) approach. EBCOT is a block-based encoder that encodes wavelet coefficients in fixed size blocks independently.

3.3 Interframe Coding

The important aspect of video compression is to exploit the temporal correlation that exists between the frames. Interframe coding (Haskell *et al.* 1997) refers to video coding that makes use of this temporal correlation.

3.3.1. Three-Dimensional Coding

Three-dimensional wavelet coding (Tham *et al.* 1998, Taubman and Zakhor 1994) is the most elegant way of interframe frame, as it does not involve the motion estimation process. It is performed by applying a 3D transform on set of images. The major drawback of this approach is that it requires huge memory in order to accommodate the large set of image sequence (Zhao *et al.* 2000). Hence is not suitable for applications with limited bandwidth and memory storage. It is also inflexible as it depends on entire image set for decoding a particular frame.

3.3.2. Motion Compensated Coding

The motion in video is exhibited by displaying a sequence of frames over a period of time to give the illusion of continuous motion. Therefore, except for certain parts, most portions of the frame have stationary information, which is similar to those of the neighbouring frames. Motion compensated coding (Ghanbari 2000) works on the principle of exploiting this temporal redundancy between the frames. It involves two processes: motion estimation and motion compensation. In *motion estimation* (ME) (Netravali and Haskell 1995), the displacement of objects within the frame is predicted using the previous reference frame. This displacement measure is used to predict the frame in the *motion compensation* (MC) process (Al-Mualla *et al.* 2002). The prediction formed using motion compensation is known as *motion compensated prediction* (MCP).

The most commonly used motion estimation approach is block based (Al-Mualla *et al.* 2002), in which the frame is divided into blocks and motion associated with each block is determined by finding a best match. The best match is found in its surrounding area, called the search area, in the reference frame. The resultant displacement measure is called a *motion vector*. This motion information is coded along with the frames.

3.3.3. Model Based Coding

Model based coding (Al-Mualla *et al.* 2002) using analysis and synthesis process to estimate the motion for video coding. This method is highly complex algorithms and computationally expensive because it requires sophisticated computer vision and

graphical tools for the coding (Pearson 1995, Eisert and Girod 1997). Each frame is broken into different objects and its motion estimation is done based on its shape, location, texture, light composition and object-specific properties. The most popular model-based coding is the object-based approach, which is used in the MPEG-4 video coding standard (ISO/IEC 14496-2:2001/Amd 2:2002). Its complexity precludes the use of object-based coding in computationally-restricted applications, and the impact of errors when handling complex scenes can be extremely severe.

4. Existing Solutions

The recent advances in networks and multimedia rich terminals pose a challenge for providing video in diverse specifications. The different approaches, which make video adaptable to the above challenges, are presented here. These techniques are broadly classified as video transcoding and scalable solutions.

4.1. Video Transcoding

Transcoding is one of the commonly used technologies to provide adaptable video to users with different terminal resources, and connection networks. A typical transcoding scenario is presented in Figure 4.1. Video transcoding is a process of converting a pre-encoded video from one format to another encoded format using a transcoder unit. The format represents any of the characteristics of video such as bit rate, spatial resolution, frame rate and content.

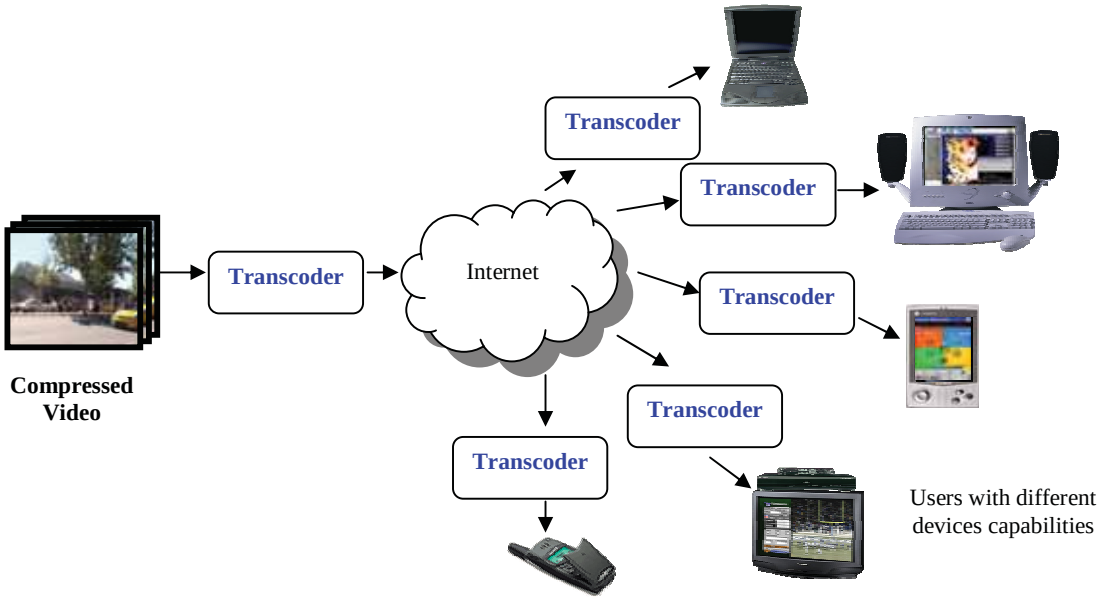


Figure 4.1: Video transcoding scenario

The transcoder consists of a decoder and an encoder unit as shown in Figure 4.2. The decoder decodes the encoded video into its original form of video and the encoder re-encodes the decoded video, the original video, into the desired format.

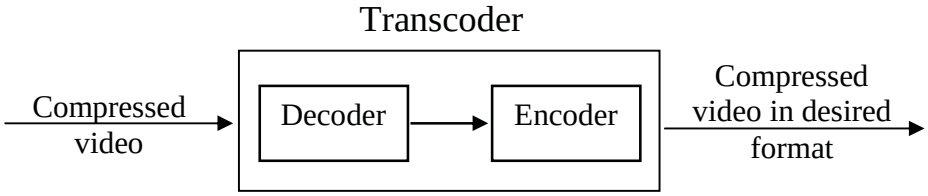


Figure 4.2: The transcoder unit

The spatial transcoding, similar to spatial video scaling, involves resizing of image resolution for video adaptation. This spatial transcoding can be achieved using one of three approaches: pixel averaging, sub-sampling or DCT decimation. Pixel averaging is the simplest approach for reducing resolution, but has lower quality due to the blurring effect of pixel averaging (Shanableh and Ghanbari 2000). The second method, which uses filtering and sub-sampling, is the most common resolution reduction technique (Mohan *et al.* 1999, Shanableh and Ghanbari 2000, Yin and Wu 2000). This approach results in a significant improvement in quality, and can be used to scale up or down in resolution. In DCT decimation approach (Zhu *et al.* 1998, Shanableh and Ghanbari 2000) has the better quality among the three it ends up fabricating blocky images consistent with the use of DCT interpolation, and also has also higher complexity (Tan and Ghanbari 1995).

Also, the motion vector estimation for a new spatial image cannot be used directly from the encoded video without some error propagation. So, motion vectors have to be recalculated using any of the following methods: random (Bjork and Christopoulos 1998), weighted average (Shen *et al.* 1999), mean (Shanableh and Ghanbari 2000, Shen *et al.* 1999) and median (Bjork and Christopoulos 1998, Xin *et al.* 2002) methods. Furthermore, all these approaches produce relatively poor results (Ahmad *et al.* 2005) due to inefficiencies or poor image matching.

Major limitations of Transcoding

The transcoder works only on pre-coded video and involves an additional encoding and decoding process, which makes it computationally expensive. In addition, the motion estimation process repeated during transcoding results in significant accumulation of

cost, computation and storage. Another disadvantage of working on pre-coded video is the violation of security when handling encrypted video. Since there is a dependence on an external source, the transcoder unit, the transcoding process is highly inflexible.

The different inter-format conversions (Vetro *et al.* 2003, Ahmad *et al.* 2005, Xin *et al.* 2005) require different transcoding architectures and procedures. Thus necessitating the transcoding process to be repeated for every single video adaptation process, like spatial, bit rate, or network adaptation as in Figure 4.1. This is a time consuming and inefficient process, which results in huge computational complexity. The problem also lies in finding an optimal transcoding approach for a given adaptation scenario.

Another major problem with transcoding is the propagation of drift error (Youn *et al.* 1999, Youn and Sun 2000, Dogan and Sadka 2002). The drift error is caused by the mismatch of reconstructed images in the encoder and decoder and when these images are used for further predictions. This causes quality degradation of the transcoded images. Additional drift is introduced by every spatial or other reduction (Yin *et al.* 2002) due to inaccurate motion vector refinements. Although there are some drift-free architectures (Yin *et al.* 2002, Sun *et al.* 1996), which reduce the quality degradation to some extent at the cost of high complexity, they are not strictly drift free.

Summing up, transcoding is a brute force way of providing video adaptation. A better alternative to transcoding is by providing video in a scalable manner.

4.2. Scalable Solution

The scalable approach provides adaptable video content that is capable of gracefully accommodating to heterogeneous networks and different user specifications. In scalable video coding, the video is encoded once but is decoded in a variety of ways. For example, by truncating particular layers or bits from the single compressed stream to achieve different video qualities, spatial resolutions and/or temporal layers for the different needs of the users or network as in Figure 4.3.

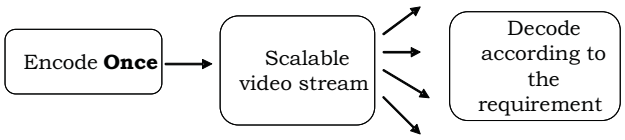


Figure 4.3: Characteristic of a scalable approach

The three major types of scalability are: quality (or data rate) scalability, spatial (or resolution) scalability, and temporal (or frame rate) scalability. In quality scalability, video images with different data rates are obtained by transmitting or processing more or fewer bits from the video data stream. The higher the number of bits, the higher is the quality of the image. Similarly, decoding lower/higher versions of the image helps achieve resolution scalability. Temporal scalability is acquired by varying the frame rate of the image.

4.2.1. Scalability in Video Standards

Scalability is not a strong element of current video coding standards like MPEG-(1, 2, 4) and H.26 (1, 3, L). The following is an overview of scalable features of the popular video coding standards.

The early forms of video standards such as MPEG-1 and H.261 do not support any scalable features as they were targeted for a single application with fixed data rate. The original version of H.264/AVC does not support any scalability, although a scalable extension (Schwarz *et al.* 2007) was appended in late 2007. The scalability offered by H.264 is similar to the other MPEG standards, which use a layered approach.

MPEG-2 Layered Scalability

The MPEG-2 standard (ISO/IEC 13818) was the first international video coding standard to provide scalable coding. It supports layered scalability, built on DCT coefficients. In the layered approach, the data is coded into one base layer and one or more enhancement layers (Haskell *et al.* 1997). The base layer is independently coded and contains video information in the coarsest form. The enhancement layers enrich the video by adding spatial, temporal and/or quality to the base layer. In case of spatial scalability, the base layer represents the lower resolution and adding more enhancement layers helps achieve

higher resolutions (Rao *et al.* 2002). Also, the enhancement layer uses prediction from the base layer without the use of motion vector. Although spatial scalability supports various resolutions, it adds coding complexity. The temporal scalability is achieved by using prediction from the base layer, similar to the spatial scalability, but with the use of motion vectors. Adding more enhancement layers helps achieve higher frame rate. SNR scalability helps achieve at least two different qualities using the same resolution frames from the base and enhancement layers. The addition of enhancement layers in the form of DCT bits improves the quality of the video. The transmission errors mostly lead to drift issues (Johnson *et al.* 1994).

MPEG-4 Fine Granular Scalability

MPEG-4 (ISO/IEC 14496-2:2001/Amd 2:2002) introduced a more flexible scalability tool that allowed scalable coding at the video object level. This scalable coding scheme is known as the Fine Granular Scalability (FGS) and has the same layered representation as MPEG-2. The FGS system (Li 2001) codes video into a base layer and few enhancement layers, but in contrast to the traditional approach, FGS uses a finer bit stream in the enhancement layers. This allows the enhancement layer to be truncated into any number of bits to provide partial or complete enhancement in proportion to the number of truncated bits. The fine scalable bit stream is achieved by bit-plane coding of the DCT coefficients in the enhancement layer. However, the standard does not specify how the truncation of bits is performed and requires additional modification of the encoding algorithm (Wang *et al.* 2001, Zhang *et al.* 2002), which results in increased complexity and quality degradation (Li 2001). Temporal scalability is achieved by combining the FGS technique with temporal scalability, Fine Granular Scalability Temporal (FGST), as in (Li 2001, van der Schaar and Radha 2001). However the scalable features of MPEG-4 are less functional than its contemporaneous image standard JPEG2000.

Limitations of layered and FGS approach

The base layers are non-scalable, that is, lower versions of the images below the level obtained from base layer are not possible. The enhancement layers are meaningless without the base layer, as they employ differential encoding with base layer as their

reference. Motion compensation is difficult to be incorporated in layered coding, as the base layer and enhancement layer tend to have different reference frames, causing drift error.

JPEG2000 and MotionJPEG2000

Although the focus here is on video coding standards, it is important to mention the state-of-the-art scalability used in the image standard, JPEG2000. JPEG2000 (Taubman and Marcellin 2002) provides a wide set of embedded scalable functions using the wavelet transform and EBCOT (Taubman 2000), and has superior performance over any current still image representation standard. Motion JPEG2000 is an extension of JPEG2000 in the video domain, therefore sharing the same properties of JPEG2000, but it has a major limitation. It does not remove temporal redundancy, which is the key compression factor in video coding, and hence has no motion compensation option. Motion JPEG2000 is used for applications where stand-alone intra-frames are critical and motion artifacts would be unacceptable, notably in surveillance video and digital cinema.

4.2.2. Wavelet Based Scalability

Recently, wavelet based video coding has been gaining importance for its excellent compression efficiency and its intrinsic scalability feature through multidimensional data presentation and the possibility of embedded representation.

3D DWT video coding

In the beginning the attractive properties of subband coding led to the development of three-dimensional coding. Karlsson and Vetterli (1987) were the first to propose the use of 3D DWT for video compression. Different variations of 3D coding were proposed, such as by Tham *et al.* (1998), and Taubman and Zakhor (1994). The main advantage of 3D DWT is its simplicity. Even though 3D coding offeres a simple scalable option it requires large buffers and incorporates very long delays. Though some architecture were proposed to tackle this problem (Xu *et al.* 2002) the early form of 3D video coding did not support motion compensation (Bosveld *et al.* 1992, Karlsson and Vetterli 1987), which was later introduced by Kronander (1989, 1990). Further investigations in motion

compensation led to many proposals including global motion compensation (Wang *et al.* 1999) and multiresolution motion compensation (Zhang and Zafar 1992).

Inter-frame motion compensated wavelet coding

A number of video coding techniques aim to provide embedded bit stream with adaptation to different spatial, quality and frame rate capabilities. One such technique is the Motion Compensated Temporal Filtering (MCTF) (Ohm 1994a, Chen and Woods 2004), which performs three-dimensional (spatio-temporal) wavelet transform along with motion compensation. It can be classified into two categories based on the order of spatial and temporal transform, t+2D (temporal-spatial) and 2D+t (spatial-temporal).

Conventionally, spatial-domain MCTF (or t+2D) (Chen and Woods 2004, Turaga and van der Schaar 2002, Secker and Taubman 2001, Bottreau *et al.* 2001) was used. It employs MCTF in the spatial domain before the spatial wavelet decomposition. Architectural aspects of t+2D and 2D+t coders can be referenced in (Ohm *et al.* 2004, Verdicchio *et al.* 2003). Although this approach has a good coding efficiency and low complexity, it has many drawbacks. It has limited spatio-temporal structures, poor motion-estimation as motion vectors are not spatially scalable in t+2D scheme and the drift due to motion reference mismatch leads to poor subsampled images (Ohm *et al.* 2004).

An alternative design for MCTF proposed by Andreopoulos *et al.* (2002, 2003), is the 2D+t or in-band MCTF approach. Here, the frames are spatially transformed and then MC temporal filtering is applied on the transformed frames. The 2D+t approach can adapt temporal filtering process in various spatial transformed subbands independently based on the spatial resolution, temporal and content characteristics (Ye and van der Schaar 2003). This approach achieves high visual quality and better performance than t+2D in terms of complexity, scalability and coding efficiency.

Though the MCTF approaches have good performance, the major limitation is the high memory demand for temporal filtering (Clerckx *et al.* 2004). Also, the performance decreases when there is poor signal correlation. In addition temporal scalability is

difficult to achieve in 3D coding and there is a limitation on the achievable temporal levels. Such schemes are not very flexible in achieving scalability modes, especially temporal, and results in degraded-quality frames (Zhao *et al.* 2000).

The popular wavelet video coding scheme, Motion Compensated Embedded Zero Block Coding (MC-EZBC), was proposed by Woods and Chen (2002). It is a fully scalable video coder which employs block-based, spatial domain MCTF to achieve improved performance. In this codec, the MCTF is first applied, followed by spatial subband/wavelet transform, and then an EZBC entropy coder. By using the MCTF, this system does not suffer the drift problem exhibited by hybrid coders. The spatial coder used here has high embedded and scalable features and capable of achieving spatial, quality and temporal scalability. The important features of this coder are that is uses sub-pixel accurate motion estimation and perfect reconstruction when no quantization is used. This codec uses hierarchical, variable-size block matching motion estimation technique. The drawback is that the motion information is coded in a non-scalable manner in bitrate or resolution (Hu *et al.* 2004, Wu and Woods 2005). Some of the embedded wavelet codecs are SPIHT (Said and Pearlman 1996, Kim and Pearlman 1997, Kim *et al.* 2000) and 3D embedded subband coding with optimal truncation (3D-ESCOT) (Xu *et al.* 2001).

JPEG2000 and its inspired scalable wavelet coders

The wavelet transform act as a suitable tool for multiresolution representation of the video signal (Mallat 1989a). This property is highly valued in achieving spatial and temporal scalability during video coding applications. Also, this feature has sparked a lot of research activities in wavelet based video coding, including MPEG standardisation (Chen and Woods 2002, Ebrahimi *et. al.* 2002) and has been used in the image coding standard JPEG2000.

State-of-the art scalability introduced in the recent image coding standard JPEG2000 (Skodras *et al.* 2001). It uses bit-plane coding to achieve both data rate and spatial scalability. In contrast to the conventional scalable coding techniques, which rely on non-scalable base layer and DCT, this coding scheme is based on the wavelet transform with

no separate base and enhancement layers. It also makes use of the embedded coding scheme, EBCOT, to provide a rich set of features such as quality scalability, resolution scalability, spatial random access and region-of-interest coding. The scalable versions of the image can be constructed by simply decimating the bit stream at any required point.

Inspired by the scalability feature of JPEG2000 many research works were developed reflecting some its features. In (Zhao *et al.* 2000), a scalable video wavelet coder is proposed which performs temporal and spatial scalability. This employs a flexible frame grouping, rather than the fixed temporal structure, to adapt to scene changes and a modified EBCOT coder. Hence it can be considered to be a differential form of JPEG2000. Motion compensation is performed in the wavelet domain using telescopic motion estimation and coded in layers. The scalability of this codec is evaluated in context of a prototype video streaming system in the form of hierarchical bit stream syntax. This framework has not fully utilized the features of wavelet transform that is multiresolution motion estimation, and uses a complex system instead involving external layering of motion information. The system on the whole is highly complex and involves huge computational expense.

Another scalable video coding was proposed by López *et al.* (2005). It provides full scalability using an in-band MCTF and EBCOT coder. As this codec is based on JPEG2000, it has some appreciable scalable properties. Additionally, it aims to remove temporal redundancy using a differential prediction-based coding along with motion compensation approach. Also, this codec tries to improve performance using some error resilience technique as in JPEG2000. Even though it achieves full scalability, it has many drawbacks. It uses a MCTF architecture, which demands huge memory and computation. The motion predictions are not performed in the wavelet domain and are not represented in the simple MRME; instead a complex structure is used. It has poor performance, much lower than the MPEG-4 standard.

In (André *et al.* 2007), the authors present a scalable-video wavelet coding, which is backward compatible to JPEG2000. It uses spatial-domain (t+2D) MCTF and the

EBCOT encoder. This codec also aims achieve an optimal algorithm for rate and quality allocation based on rate-distortion curves. The use of a non-scalable motion vector is a major drawback, as it greatly lowers the performance of the coder and degrades the quality due to the misrepresentation of motion information. This scalable coder has a lower performance than H.264.

4.3. Overview of Our Approach

In our work, we aim to develop a scalable video coding framework which has similar scalable properties as JPEG2000, though at this stage we only concentrate on achieving spatial scalability. For this purpose, we want to effectively use the features of the wavelet transform. We use 2D wavelet transform and perform the motion compensation process in the wavelet domain, using scalable motion information through multiresolution motion estimation. In addition, we have presented a framework, which provides selective transmission of data based on the network constraints and user requirements.

5. Methods and Simulations

This chapter gives a detailed description of all of the approaches and trials experimented leading up to the scalable selective data transmission framework, which are organised into two sections. First the initial investigations, which involve all the methods and trials experimented detailing its purpose and efficiency with respect to the current research objective, and gradually progressing to the next section, which gives a comprehensive account of the final adopted scalable methodology and its selective data transmission representation.

5.1 Initial Investigations and Trials

Our initial objective was to develop a scalable video coding framework, which could incorporate the following features,

- Spatial Scaling
- Quality Scaling, and
- Temporal Scaling.

In order to achieve a scalable bit stream, a wavelet transform representation was our principal choice, as validated in the Literature Review, coupled with multiresolution motion estimation. The key problems encountered in the use of multiresolution motion estimation for scaling purpose are error propagation due to scaling of motion parameters across different resolution levels, the identification of such errors in motion blocks and its appropriate error handlers. A representation of such a scalable framework is given below in Figure 5.1. Additionally, we wanted to control the data rate depending upon network constraints such as the available bandwidth and user preferences. This would allow the data rate to be constrained as needed, reducing visual quality but allowing video transmission nevertheless to proceed. Though we initially experimented with full scalable video coding we had mainly focused on spatial scaling. Other such experiments will be detailed in this section while the final approach will be clearly presented in the next section.

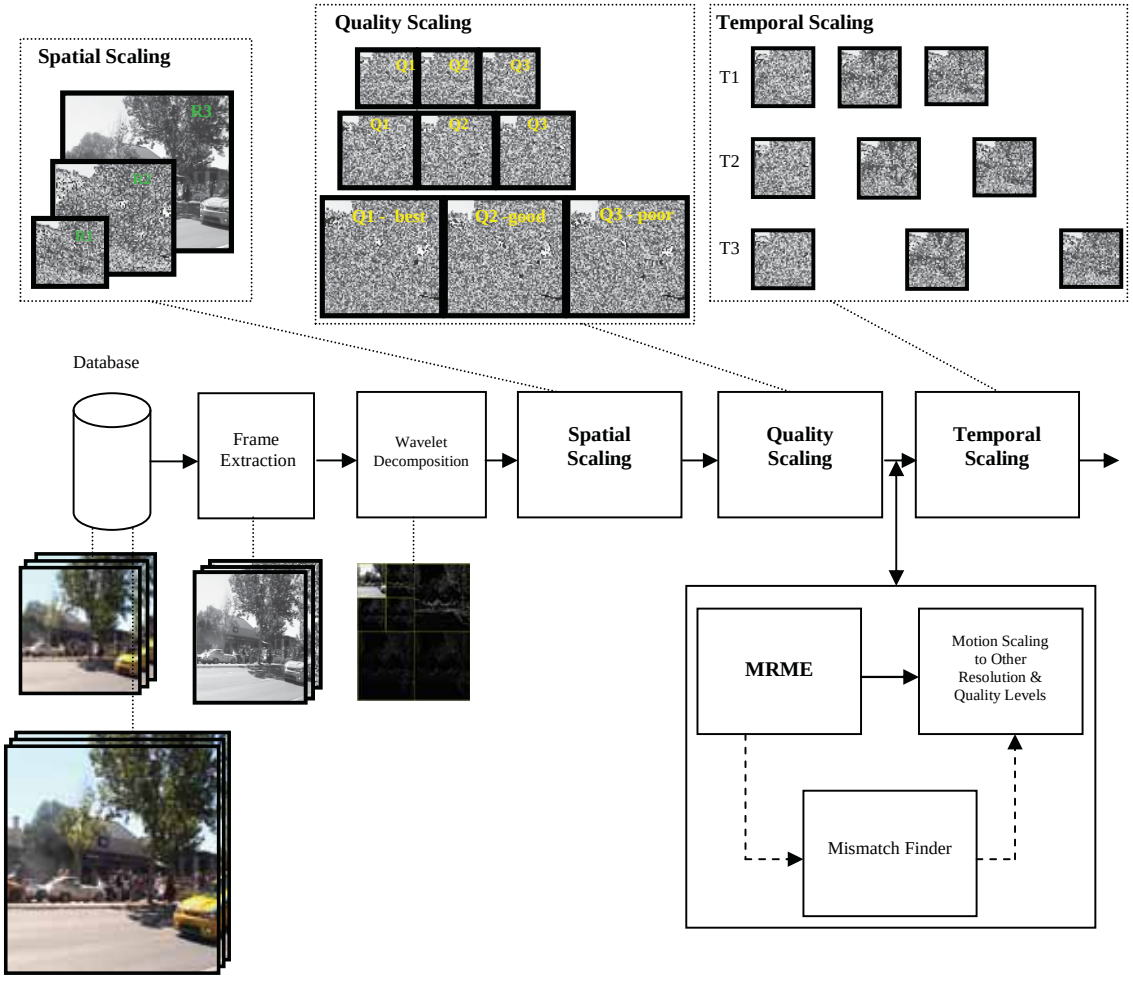


Figure 5.1: Fully scalable video coding framework

5.1.1 Initialisations of General Parameters

Sample Video Data

Video samples were shot specifically for this research. Three sample videos were taken covering the basic aspects of video diversity, as discussed in Chapter 6.

Simulation Environments

All the experiments were simulated in *MATLAB*[3]. MATLAB is a wonderful platform for developing applications and functions that uses complex mathematics, and supports excellent tools to handle images, video and wavelets. Hence it was the appropriate choice for our experiments. As MATLAB can only handle video in uncompressed avi format, the video frames were extracted externally using Adobe Premiere Pro[4]. Video frames of resolution 720x576 were extracted from the video samples and stored in bitmap (bmp) format. We restrict our image resolutions *(MxN)* in the order of $(2^{2m}x2^{2n})$, as it is easy to work with images that are in powers of two especially in a multiresolution structure, although the work can be generalised to other resolutions.

The imported images were then converted to grey scale from RGB. Conversion to grey scale helps to reduce the computation load as processing is carried out in only one plane and because grey scale is the best choice for representing the three dimensional space of colours.

5.1.2. Multi-Level Content Extraction

Achieving complete scalability using only a single source requires the representation and manipulation of the data in various forms. That is video has to be represented in different level of resolutions for spatial scalability and coarseness for quality scalability. Therefore, a multi-level representation of video based on the required contents, such as resolution or quality, is needed before processing them for scalable functions. This can be done in various ways:

[3] MATLAB, an acronym for *matrix laboratory*, is a high-level language for technical computing.
[4] Video editing software

(1) by simply resizing the image into the different resolution sets, or

(2) using any image coders capable of producing images in different specifications, or

(3) using wavelet transform

Although wavelet transform was our principal choice we also considered the other alternatives.

5.1.2.1 External Source

This is a very simple approach, as the work of performing multi-level representation is outsourced and involves no wavelets. This was done to get an overall idea of the entire research and to better understand the requirement of this phase. Adobe Premiere Pro was used for the extraction of images in different resolutions for testing resolution scalability. So initially, images of resolution 88x72, 176x144 and 352x288 were extracted from the software and sent to the video processor for further operations.

5.1.2.2 JPEG2000 Coder

As JPEG2000 was our initial inspiration we used a JPEG2000 encoder/decoder to have a better understanding of its mechanism. The JPEG2000 coder is built on wavelet transform. Thus it is similar to the use of wavelets.

The freely available Kakadu JPEG2000 coder v5.0, developed by David Taubman (2001), was used. It scales the images based on different parameters, like resolution, quality and rate. Two sets of images were extracted using this coder, one based on resolution, and another on quality. The scaling factor was a power of two. So if the frames were reduced at the rate of 50%, 25%, 12.5% and 6.25% then it represents the images in decreasing levels of quality, Figure 5.2 (Sorell 2005). If the frames are coded at the rate of r1, r2, and r3, then the resolution of the images are decreased from 704x576 to 352x288, 176x144, and 88x72, Figure 5.3. The resolution 88x72 being so small, was later removed out of testing data and only two levels of resolutions excluding the resolution level 704x576 were used.

NOTE:
This figure is included on page 37
of the print copy of the thesis held in
the University of Adelaide Library.

Figure 5.2: Quality scaled images using JPEG2000 coder

Figure 5.3: Resolution scaled images using JPEG2000 coder

Advantage

The benefit of using a JPEG2000 coded image set is that the images share the same properties of a wavelet transform and that of JPEG2000. This results in our image samples being wavelet-transformed images.

Disadvantage

On the other hand, if the wavelet transform is applied directly rather than achieving the same through the JPEG2000 coder, the images have greater flexibility. There would also be no limitation on level of decompositions, use of wavelets filters and mainly on reusability of motion parameters. Using a coder is the equivalent of using a transcoder. So wavelet transform is much preferred than the use of any coder even JPEG2000 coder.

5.1.2.3. Wavelet Transform

In recent years, wavelet transform has emerged as an efficient tool for the representation of image and video in hierarchical form (Mallat 1989a). The wavelet transform is used for representing images or video in different levels of resolution and quality, as evident from its application in JPEG2000 (Taubman and Marcellin 2002) and Motion JPEG2000. This multi-level representation of content helps to achieve scalability of video among different user devices and network conditions, and as the original image is divided into a number of subimages each with different resolution and quality, it gives great flexibility and ease to code and process these images. It has also been demonstrated that discrete wavelet transform provides better performance than Fourier or Cosine transforms (Woods and O'Neil 1986, Vetterli 1984, Mallat 1989a, and Xiong *et al.* 1999).

5.1.2.3.1. Choice of Wavelets

After the extraction of video frames from the database, the frames are subjected to wavelet decomposition. The simulation was initially tested using 'bior' and 'haar'. Haar is the simplest form of wavelet while bior is a biorthogonal wavelet with similar properties of Daubechies 9/7 wavelet used in JPEG2000. Biorthogonal and Daubechies wavelets have better performance (Topiwala 1998); the Haar wavelet was chosen for its simplicity.

The images can be subjected to 2D wavelets along with motion estimation predication or using 3D wavelets, when considering video as a three dimensional array of data. 3D wavelet based approach, which has been extensively researched (Topiwala 1998), is not suitable for video signals as video signals are best represented by 2D projections due to

its inherent ability to effectively model the temporal redundancy (Nosratinia 1995, Nosratinia *et al.* 1999). 3D wavelets are also computationally complex; hence we have adopted 2D wavelet approach in our work.

5.1.2.3.2. Wavelet Decomposition

When the two-dimensional (2D) wavelet transform is applied to video signals, the video image is decomposed into one Approximation subimage and three Detail subimages (horizontal, vertical and diagonal images) at the each level. The approximation containing most of the energy can further be decomposed. In our work the video frame is decomposed up to two levels, as shown below.

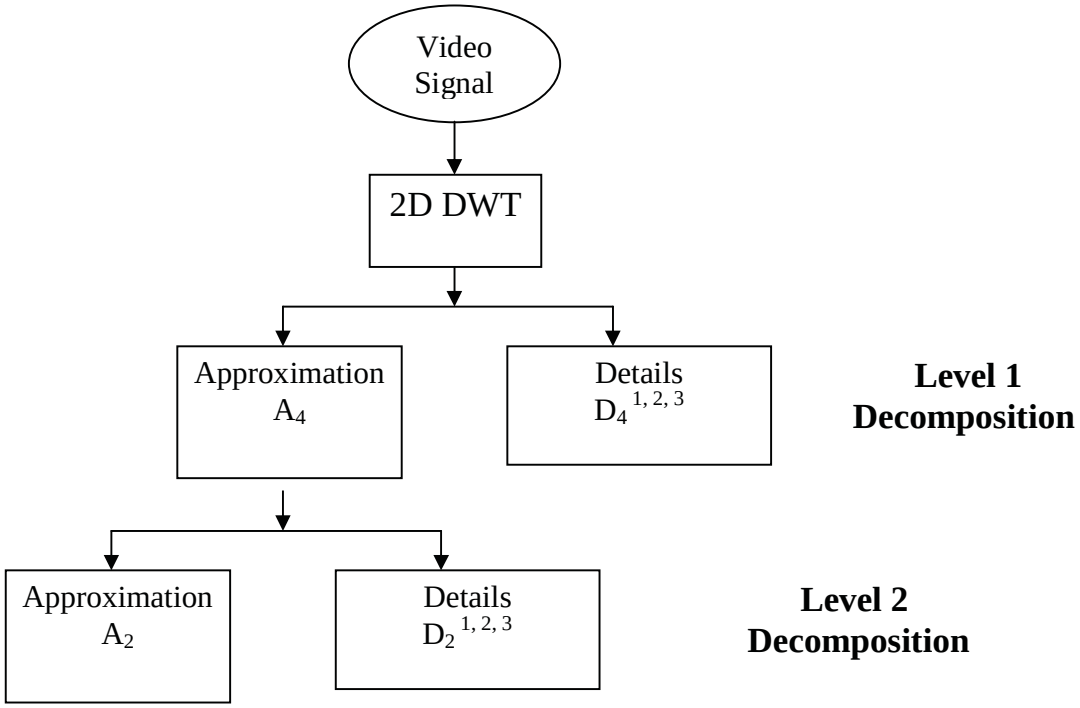


Figure 5.4: Two level wavelet decomposition of video signals

The two dimensional (2D) discrete wavelet transform (DWT) of video image *f(x,y)* of resolution depth *m* (here *m* is 2) can be represented by

$$\{ A_{2^1} f, \left[D^j_{2^1} f\right]_{j=1,2,3}, \cdots, \left[D^j_{2^m} f\right]_{j=1,2,3}\} \quad (5.1)$$

where, A denotes the approximation coefficients and D^j (j=1,2,3) denotes details coefficients (horizontal, vertical and diagonal respectively).

As a result of wavelet decomposition, a total of 7 subimages are obtained with four subimages at level 2 and three subimages at level 1 (Figure 5.5). The subimages thus obtained from wavelet decomposition can be represented in a pyramidal structure as shown in Figure 5.5. This pyramidal structure is helpful to represent video in different resolutions and with different levels of quality from the same source, which are further explained below.

Figure 5.5: Pyramid structure of wavelet decomposition and reconstruction (Mohan *et al.* 1999).

5.1.2.3.3. Multiresolution Representation

In our work, as there are only two decomposition levels, we have resolutions 144x176 (R1), 288x252 (R2) and 576x704 (R3), and their approximations are represented by subimages A_2, A_4 and A_8 respectively, shown in Figure 5.6. The 144x176 images are represented by the subimages at Level 2 and resolution 288x352 at Level 1. The approximation image A_4 is obtained by constructing the subimages at Level 1 and similarly approximation A_8 is constructed by combining subimages at Level 1 and the previously constructed approximated A_4 image, which is clearly depicted in Figure 5.5.

Images with different resolutions can thus be obtained by constructing images at each level. This provides easy access to images with different resolutions for applications that require images in different sizes.

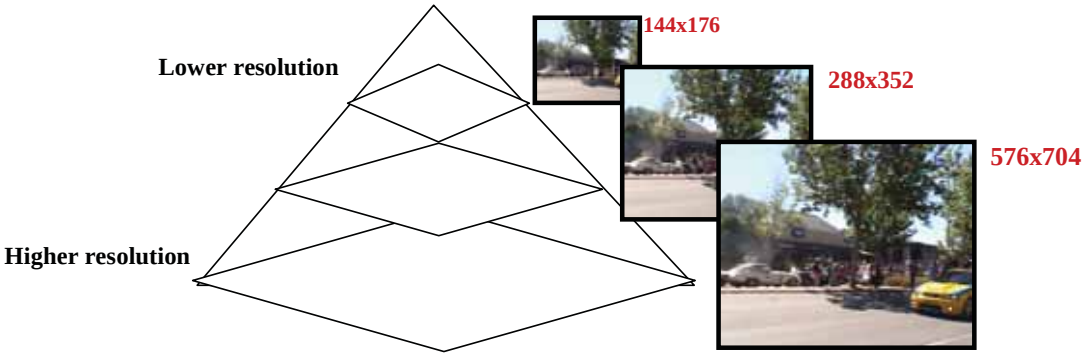


Figure 5.6: Multiresolution representation of wavelet decomposed images

5.1.2.3.4. Quality Representations

As mentioned earlier, it is possible to obtain different quality images from the same resolution level either by the addition or the removal of details from the images. This motivated us to experiment with quality scalability. Thus for our experiments we used three quality levels, each with varying amounts of details, Figure 5.7. They are

- Q1: high quality images with maximum details (100% wavelet coefficients)
- Q2: moderate quality with lesser details (50% coefficients), and finally
- Q3: poor quality with fewer details (25% coefficients)

(a) High quality (Q1) (b) Moderate quality (Q2) (c) Low quality (Q3)

Figure 5.7: Different quality levels of Resolution 1 (144x176)

The details correspond to horizontal, vertical and diagonal coefficients. Upon removal of these coefficients either from the same level or from the preceding levels, the quality level of the image can be varied. Since these quality variations are obtained from the

same wavelet pyramid structure, they are highly correlated with the resolution images, thereby making the implementation of motion predictions much easier.

5.1.2.4. Motion Estimation

There are different ways of performing motion estimations: block-matching, pixel-based, gradient approach, frequency based approach and so on (Al-Mualla *et al.* 2002). Here, motion estimation was performed using the *block-matching scheme* due to its simplicity and cost efficiency (Rhee *et al.* 1995). In this approach the entire video frame is divided into a number of small blocks and each block is associated with a single motion vector. In the block-matching scheme, each block in the current frame is matched to its corresponding block and its neighbors, in the previous reference frame, to determine an appropriate match denoting the movement of the block.

Block-Matching Algorithm

Broadly speaking, the block-matching algorithm can be classified into two: full search and fast search algorithms. The full search, also known as exhaustive search, searches the entire search area at every single location to find a match for the block. As a result it gives a best possible match compared to the others but it is also computationally expensive. Fast search algorithms, as the name suggests, try to achieve the similar result by searching only at selected locations in the search area.

Initially the simulations were conducted using both exhaustive search and fast search. In fast search approach, two-dimensional log search and diamond search (Al-Mualla *et al.* 2002) were used for the simulation. As the accuracy of motion estimation was vital for our simulation, we chose exhaustive search over fast search.

Block-Matching Function

There are various block-matching functions, of which the most commonly used Sum of Absolute Difference (SAD) function is used as the block-matching criteria, which is represented by

$$SAD(x,y) = \sum_{i,j \epsilon \Omega} |I_t(x,y) - I_{t-1}(x+i, y+j)| \tag{5.2}$$

The position of the block (B) at the location *(x, y)* in current frame I_t is denoted by B_t and $I_t(x,y)$ is the gray scale representation of the block B_t. Using the matching criteria, SAD, the displacement of this block with respect to the previous frame I_{t-1} is found in terms of a vector V_i *(x, y)*, denoting the horizontal and vertical displacement of the block with minimal error, in a search area Ω surrounding the block in the previous frame. The SAD process is a simple and computationally less expensive one. It calculates the absolute difference of each block in the search area Ω and chooses an appropriate block with the lowest value. The process of block-matching scheme is represented in Figure 5.8.

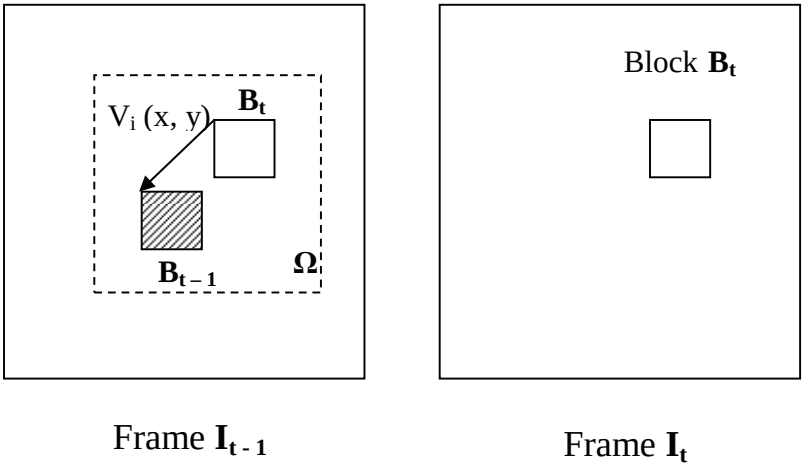


Figure 5.8: Block-matching motion estimation scheme

Block Size

Initially we used a standard size of 8x8 pixels for the blocks. This block size was used across all the resolution. This was appropriate for images with resolution 88x72 but proved to be ineffective for higher resolutions. Immediately we knew that it was inappropriate to use the same size blocks across all resolutions. Therefore the block should not be either too small or too large in respect to the resolution of the frame. We kept on experimenting with different sizes until we came up with an optimal solution. For our work we standardized the block size as 16x16 for resolution (R1) 144x176 and for higher resolutions, the block size increased in same the ratio as of the resolution, ensuring the same number of motion vectors regardless of resolution.

Search Area

As the search area differs with respect to the block size and the motion involved (Zafar *et al.* 1993), the experimentation went through the same cycle as block size trials. If both block size and search area parameters were to change it would be difficult to identify which of the two parameters caused the desired result, so for most of the experiments the search area factor was kept constant. The search area should vary in accordance with the block size, as they are both highly correlated. Thus the search parameter, *p*, initially, started out with 8 pixels around the block, as shown in Figure 5.9.

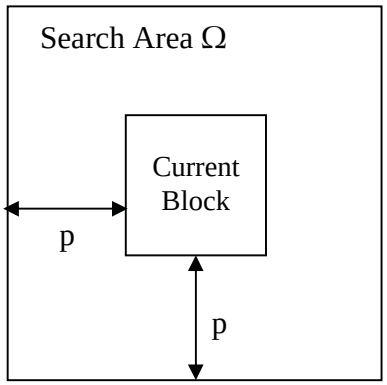


Figure 5.9: Search area and current prediction block

The search area should also vary with resolution and change in motion. Even though consecutive frames were used, there would also be considerable amount of motion in the frames as in the sample videos. Hence, the search parameter was made equal to the size of the block.

Frame Types

In Motion JPEG2000 intra frame coding, i.e. only I frame, is used. In order to take advantage of temporal redundancy, P frames (Predictive coded frames) are used in addition to I frames. B frames (Bi-directionally predicted frames) are not used at this stage.

5.1.2.4.1 Multiresolution Motion Estimation

Motion involved in the different layers of pyramid is different but is highly correlated as they represent the same motion pattern but at different sizes (resolutions) and representation (details and approximations). Multiresolution motion estimation exploits

this motion correlation among the different layers in order to reduce the computational complexity of motion estimation process, which is otherwise performed at every single layer.

Multiresolution motion estimation can be performed by two different ways: *bottom-up* and *top-down.* In the bottom-up approach (Rhee *et al.* 1995, Conklin and Hemami 1997, Kim *et al.* 1998) the motion estimate is initially performed at the bottom of the pyramid, on the higher resolution images, and the resultant motion vectors are used as the basis for other resolutions up the pyramid, whereas in the top-down approach (Tzovaras *et. al.* 1994, Mallat 1989a) the initial motion estimate is performed on the lower resolution images. We initially started with the bottom-up approach as we thought the motion estimates would be more precise if done in a higher resolution image but it turned out to be an expensive and redundant process. Performing motion estimation on higher resolution images meant working on a large volume of data when similar results could be obtained with very little data by working on lower resolution images top-down. In the top-down approach, the lower resolution image, at the top of the pyramid, here subimage A_2, contains the larger percentage of the total energy and has the lowest frequency band. This has more information details and provides better motion estimation with less data. Hence the *top-down approach* was used for motion estimation.

As mentioned earlier, we originally used constant-size blocks for the multiresolution motion estimation. Its downside, especially in multiresolution motion estimation, was that it was difficult to translate the motions accurately to other resolutions because the corresponding blocks did not share the same motion. So we used variable-size blocks for multiresolution motion estimation, which is explained in the next section.

Variable Block-Size Motion Estimation

In variable block-size approach, which was primarily proposed by Zhang and Zafar (1992), the size of the block differs as the resolutions of the frames vary. The size of the block can be defined as

$$\text{Block Size} = q.\ 2^{D-L} \tag{5.3}$$

where,

q is the block size at the lower resolution frame

D denotes the total number of wavelet decompositions

L denotes the decomposition level

As the total number of decomposition used in this research is 2, the block size becomes $q \times 2^{2-L}$. The value of parameter q is kept constant and takes the value 16 in this work. The change in block size with the change in resolution keeps the total number of blocks in each resolution a constant, as shown in Figure 5.10. This simplifies motion vector translation across the different resolution level. Since the total number of blocks is the same for each resolution level, the motion within the blocks also represents the same part of the image in contrast to the constant block-size method (Uz *et al.* 1991) where the blocks across the resolution represent different motion.

Figure 5.10: Variable block-size motion estimation (Zhang and Zafar 1992)

As mentioned earlier, the motion at lower resolution forms the initial basis for predicting the motion at next higher resolution but with some refinements. In the variable block size approach the motion vector translation is represented as

$$\text{Motion vector} = \textbf{scaling} + \textbf{refinement} \qquad (5.4)$$

$$\textit{i.e.} \quad V_h(x, y) = 2^{2-L} V_l(x, y) + \Delta(x, y)$$

where,

V_l (x,y) = the motion vector at lowest resolution of the decomposition

V_h (x,y) = the motion vector of next higher resolution, here at Level 1 and 0

Δ (x, y) = motion vector correction

At subimage A_2, which the lowest resolution image, a full search motion estimation is performed and the resultant motion vectors are denotes as V_l (x,y). Then the motion vector from the lowest resolution level is scaled for use at higher resolution. The scaling factor is 2^{2-L} times the lowest resolution motion vector.

Even though motions in the multiresolution decompositions are highly correlated they have minor variations due to representation of the image in the different frequency levels (horizontal, vertical and diagonal subimages) as shown in Figure 5.10. The motion vector representation across the different subimages is given by

$$V_h^j(x, y) = 2^{2-L} V_l^j(x, y) + \Delta(x, y) \qquad (5.5)$$
$$for\, j = 1, 2, 3,\ h = 4, 2, and\ L = 1, 0$$

where,

j = denotes the subimages 1, 2, 3 (horizontal, vertical and diagonal respectively)

V_l (x,y) = the motion vector at lowest resolution of the decomposition, V_2

V_h (x,y) = the motion vector of next higher resolution, here at Level 1 and 0, V_4 and V_8 respectively

L denotes the decomposition level 1 and 0

In the beginning, the simulations were performed using all the detail subimages (horizontal, vertical and diagonal images) and with different combinations of all the detail subimages. The usage of these subimages for the purpose of motion estimation requires more computation. Although the use of details, D_i subimages, along with approximation, A_i subimage, provides good accurate estimates (Zhang and Zafar 1992, Zafar *et al.* 1993), it is done at the cost of having large computational overhead. By using only the approximate image for basic motion estimation, most of the image energy is concentrated. The resultant motion now becomes,

$$V_h^0(x,y) = 2^{2-L} V_8^0(x,y) + \triangle(x,y) \quad for\ h = 4,2, and\ L = 1,0 \tag{5.6}$$

where,

$V_8{}^0(x, y)$ - denotes the approximation coefficients of wavelet decomposition at Level 2, subimage A_8

$V_h{}^0(x, y)$ - denotes the approximation coefficients of wavelet decomposition at Level 1 and 0, subimage A_4 and A_2 respectively

5.1.2.5. Error Correction

The discrepancy caused by motion translation across different resolutions results in translation error, which is handled by incorporating a motion correction factor Δ (x, y). Many MRME error correction schemes have been proposed (Mandal *et al.* 1996, Kim *et al.* 1998). In (Zafar *et al.* 1993), motion refinement is applied to all the motion predictions of the frame by repeating the motion estimation process in a reduced search area using a sub-pixel approach. Although this approach gives better results it is done at the cost of having high computational expense for little gain due to the fact that the refinement process is carried out for every single motion vector. Refinement would be a great advantage if it were applied only to the necessary mismatch blocks. The use of a motion correction factor for all the blocks renders this approach a costly setback in terms of computation and coding efficiency. Another technique that uses a similar approach of Zafar et al. is the MRME indexing proposed by Zan *et al.* (2006). Here in the refinement process, the sum of absolute difference of each block is compared to the sum of absolute values of wavelet coefficients, and if it is greater then the blocks are intracoded. Although

the refinement process is carried out only for certain blocks which are higher than the threshold this approach does not give a satisfactory solution as it is performs badly at higher bit rate and has a lower performance than (Zafar *et al.* 1993).

Error correction involves two processes: identification of the mismatched blocks and the correction of the error. Many approaches were tried, to identify and correct the errors, until an efficient solution was found. Some of these trial-methods are presented here.

5.1.2.5.1. Error Identification Methods

Error identification was performed at the block level, and the error magnitude of each block was measured using sum of absolute difference (SAD) measure.

Experiment 1: Threshold

In this approach, the error magnitude of every block is compared against a threshold; if the error metric of the block is greater than the threshold then it is marked as a mismatch and separated for error handling. Choosing an appropriate threshold, which could be an isolation point between the block as good and bad prediction, was a very difficult process. Different parameters, like motion error index and scene change parameter were tested for threshold, but in vain. This approach failed because the selected threshold pointers were not able to correctly detect the error blocks, as it was not possible to set a static threshold.

Experiment 2: Displacement magnitude

Here, the error identification was based on the displacement index. This testing was done to investigate whether the displacement of a block could be used as an error pointer. As consecutive frames and slow motion sequence were used for this testing, the motion displacement between the frames was very small. In this scenario, if the motion prediction of the approximated image were to have a higher value, that is, if the block were displaced farther away from its original point with respect to its position in the reference frame, then the corresponding block was considered a mismatch. The performance of this approach was not satisfactory, even though it had a good outcome in

a slow motion sequence. This is a brute force approach and would suffer greatly when the sequence has complex motion. It is also extremely difficult to differentiate the error caused by the motion translation of MRME approach and the displacement error caused by the general motion of the sequence.

Experiment 3: Neighbouring blocks

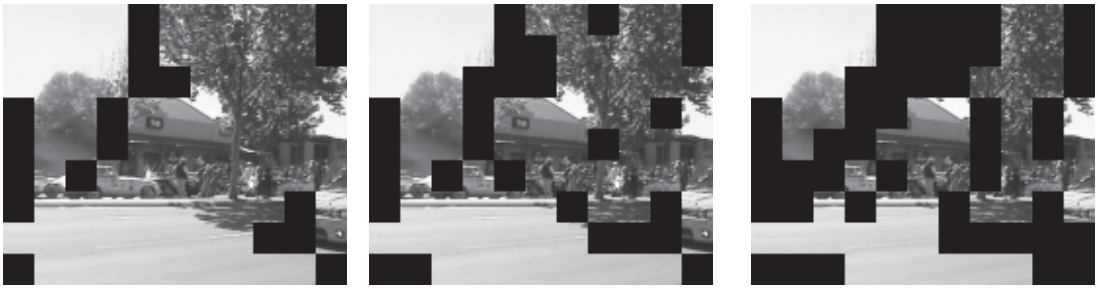
In this approach, the blocks were compared against its neighbouring blocks to determine if it shared the same motion prediction and error magnitude. This comparison was not good enough, as the surrounding blocks may not share the similar scene change. Besides, this might also lead to additional drift error, if the neighbouring blocks were also mismatches.

Experiment 4: Prediction image

The advantage of performing error correction at the sender side is that the encoder has access to all the image sets. So when the motion compensated image is predicted it can be compared to the original prediction image, and identify the blocks that have a prediction error. But this required dependence on the entire image set and introduced excessive complexity.

Experiment 5: Local motion

In this approach, the prediction error is identified based on local motion within in the frame. After repeated experiments it became clear how every block in the frame shares the local motion. Based on this information, the mismatched blocks are identified using a threshold. The blocks which fall under the threshold are identified as mismatches and require correction. The threshold can be varied, if needed, to include more or fewer mismatches as in Figure 5.11. The only drawback is that the motion considered is only local and therefore needs to include the global motion and further refinement for a better estimate. This approach forms the basis of the selective data transmission and error correction approach proposed in the Sections 5.2.4 and 5.2.5.



(a) Threshold level 1 (b) Threshold level 2 (c) Threshold level 3

Figure 5.11: Mismatch identification based on local motion

5.1.2.5.2. Error Handlers

The different ways of correcting the error blocks are discussed here.

Replacement method 1: Prediction from different resolution level

As a multiresolution representation is used here, the different resolution images in the pyramid structure share similar motion representation. This property is used here for handling the prediction error and this approach was inspired by the error correction approach used in (Kim *et al.* 1998, Zan *et al.* 2006). The motion prediction is compared along different resolution level, Figure 5.12. The prediction block that has lower error magnitude, or energy, among the different resolution level is used as a replacement for the error block. So the resultant predicted frame contains blocks from different, higher resolution levels. There are some drawbacks in this approach. It uses complex functions, intense computation, and requires a very large bit rate as the mismatches contain data from higher resolution. There is some quality degradation around the replaced blocks.

Figure 5.12: Prediction comparison along different resolution levels

Replacement method 2: Motion vector replacement

The motion vector of the error block is updated with new motion vector, obtained by repeating the motion estimation process. As the initial estimate is taken from the lower resolution image, it is not used for the second prediction. The motion estimation is performed in the corresponding higher resolution level. The new motion vectors do not give a good prediction, mainly due to the error propagation of using different prediction sets from different resolution levels.

5.1.3. Scalable Framework

In this work, we started with the aim of achieve a scalable video coding, and have studied and stimulated the suitable options leading to a scalable architecture. Even though we limited our scalable feature to spatial scalability, we managed to experiment on quality and temporal scalability to a certain extent. An overview on all three scalable features is given here.

Resolution scalability

A detailed representation of our spatial scalable framework is presented in the Section 5.2. Through the investigations and trials discussed here, we have used 2D wavelet transform and motion compensation involving multiresolution motion estimation to achieve scalability.

Quality scalability

The quality scalability is performed after the spatial decompositions. We experimented with three levels of quality scalability using wavelet coefficients as explained in Section 5.1.2.3.4. Therefore in our trials we expected to achieve different levels of quality images by increasing the details of the image. In addition, during the error correction process, further quality can be achieved for the error blocks by sending more corrected details for those blocks. Thus, in this experiment we were able to obtain different quality levels in different resolutions.

Temporal scalability

All the scalable functions are performed in the wavelet domain and temporal scalability was done after spatial scalability and quality scalability, similar to the in-band approach. Since motion compensation was done using MRME, the motion information was scalable and available to all the different levels. As a simple temporal structure was tested, the motion information was not scalable for all the different temporal levels. In our experiment different temporal levels were formed by just dropping frame, and it was designed to have three resolution levels and three quality levels for each temporal level, performing motion estimation only once. We had combined motion compensation with this design but did not at that stage perform selective data transmission. It was just an experimental stage and requires more work to deliver a complete scalable framework.

5.2 Our Approach

After investigating and testing possible alternatives, we have come up with the following framework, which provides scalable video, mainly spatial scalability, and is capable of controlling the amount of data transmitted according to the network and user constraints. The configuration of our spatial scalable framework is explained here. The design overview of this scalable framework is shown in Figure 5.13.

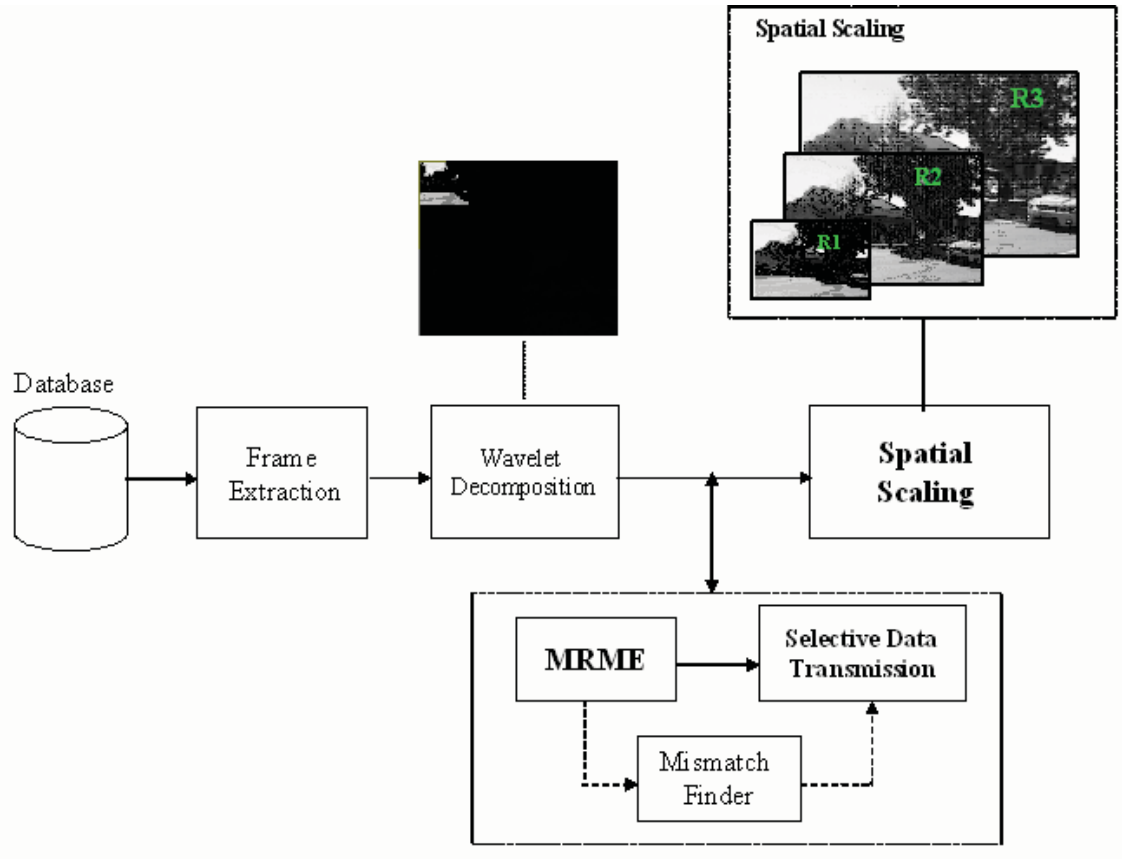


Figure 5.13: Flow diagram of our spatial scalable framework

First, a brief outline of the components to the point of motion estimation will be presented as they have been already dealt in detail in the previous section. The error correction and the selective data transmission will then be discussed at length.

5.2.1 Simulation Assumptions

Video samples of different motion types were used to study the effect of our framework in different motion sequences. The resolutions of the video images were 576x704 to facilitate easy resolution decomposition of the order of 2^{2m} x 2^{2n}.

5.2.2. Wavelet Transform

Haar wavelets were used throughout for simplicity. Every frame was subjected to wavelet decomposition. The decomposition level was set to two. Therefore three resolutions were extracted: 144x176, 288x352 and 576x704 (see Section 5.1.2.3.3).

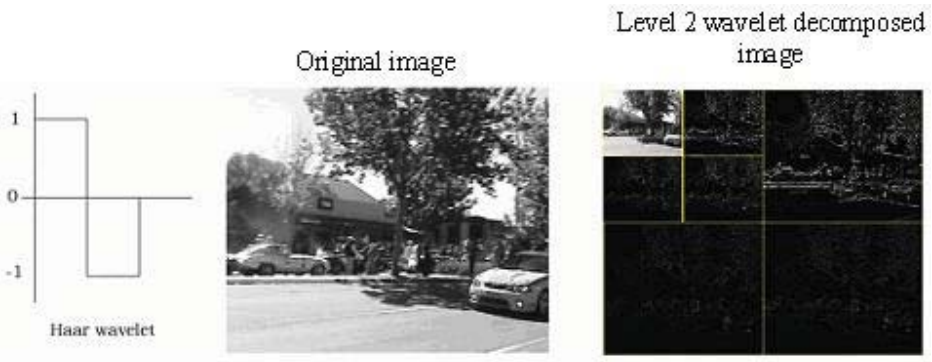


Figure 5.14: Level 2 decomposition using haar wavelet on the sample image

5.2.3 Multiresolution Motion Estimation

The multiresolution motion estimate is used to perform motion prediction on the three different resolution levels. The commonly used block-matching scheme is used for our full search with SAD as the matching criterion (Al-Mualla *et al.* 2002). We have used top-down MRME approach and have implemented variable block size method, starting with block size of 16x16 in the lower resolution (R1) and increasing in the order of 2^{2m} in a search area of p=8. A more detailed explanation is given in Section 5.1.2.4.

In our approach, the motion estimate is initially performed at the lower resolution (R1) images and the results are scaled as prediction for the other two resolutions (R2 and R3). Some of these scaled predictions have translation error and need to be corrected.

5.2.4 Error Identification

The translation of motion vectors from lower resolution to higher resolution introduces error or mismatches. This error does not occur in all the blocks, but occurs only in blocks showing significant variation in motion. Therefore proper care should be taken to correctly identify only these mismatched blocks.

In order to identify the mismatches correctly, a study of the motion pattern of the blocks in relation to the local and global motion was performed. First the error magnitude for every single block in the frame was measured using Sum of Absolute Difference (SAD) measure. These values were then mapped against the corresponding block to study their behaviour in local motion as in Figure 5.15. From the Figure, it can be seen that the error magnitude plot, blue lines, shows huge variations making it difficult to accurately pin point the error blocks. Therefore the mismatches, red markers, were plotted on top of the SAD plot and it clearly shows that the mismatches align exactly with the peak values, inferring that the mismatch blocks have higher error magnitude than the others. However, on a closer look it shows that not all these mismatches are to be replaced. So the blocks have to be further filtered within this range. To identify the blocks with the worse match, two new parameters are introduced: (1) Threshold and (2) Worst blocks, as shown in Figure 5.16 and 5.17.

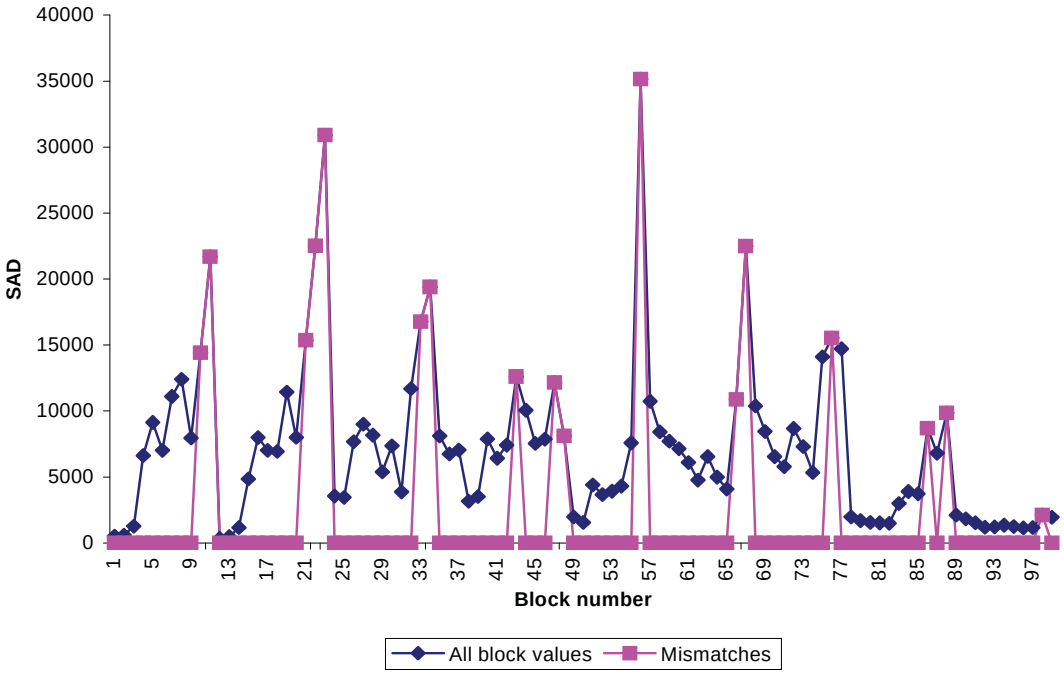


Figure 5.15: Identification of mismatch blocks

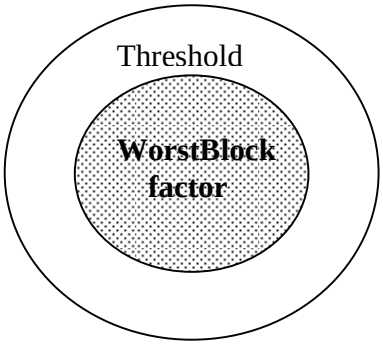


Figure 5.16: Representation of Threshold and WorstBlock factor

Threshold

This parameter helps to identify the threshold limit above which the block replacements are necessary. This is marked as the horizontal line in Figure 5.17. It generally takes an optimal value for a given motion sequence, however, it can be varied to suit the user requirement. The Threshold is defined as,

$$\text{Threshold} = \text{X}\% * \frac{1}{N}\sum_{i=1}^{N} maxErrorMagnitude_i \qquad (5.7)$$

where,

N – Total number of frames in a sequence

As mentioned earlier mismatches are the blocks with higher error magnitude, so our threshold limit is based on the error magnitude. We take into account the error magnitude of all the frames to have a perspective of global motion. The Threshold parameter can be varied by varying the value of X. That is, if X is 0 then all the blocks are considered for replacement which is suitable for higher quality video delivery; if X is 100% no replacement takes place. Otherwise X takes a pre-determined value in all other circumstances to meet some quality requirement.

WorstBlock Factor

The WorstBlock factor helps to identify the worst mismatches within the Threshold limit, represented as the vertical line in Figure 5.17. It can be defined as,

$$WorstBlockfactor = Y\% * Blocks_{(>Threshold)} \quad (5.8)$$

The WorstBlock factor can be varied, similar to the Threshold, based on the bandwidth requirements. If Y takes a maximum value, then all the blocks within the Threshold limit are replaced, that is during higher bandwidth scenario, or if Y takes a minimum value only the worst mismatches within the Threshold limit are replacement. Generally WorstBlock factor is optimised to meet a bandwidth constraint.

These two parameters are not only used in error identification but also in selective data transmission, which is discussed in the next section.

5.2.5 Selective Data Transmission

The purpose of selective data transmission is to control the amount of data transmitted based on the available bandwidth and user requirements. The amount of data, that is the block updates, is controlled by varying the two parameters: Threshold and WorstBlock factor. The working of selective data transmission is presented in Figure 5.17.

In the Figure, all blocks in a sample sequence are mapped in the order of their error magnitude. The two control parameters, Threshold and WorstBlock factor, are

represented by the horizontal and vertical line respectively. The Threshold is the deciding factor for determining the level above which block updates are essential. This Threshold helps to control the quality of the video. Low quality video is obtained by limiting the block updates to a minimum number by setting the Threshold limit to a maximum value. Similarly, by setting the Threshold to a bare minimum level the quality of the video can be increased greatly as it allows a large number of block to be updated.

While the Threshold controls the overall block-update level, the WorstBlock factor helps to further limit the number of blocks within the Threshold level by choosing only those worst mismatch blocks that definitely need to be replaced. The WorstBlock factor characterises the bandwidth limitation. If the available bandwidth is low the WorstBlock factor takes a minimum value and allowing only a few worst mismatches to be updated. On the other hand, if there is no bandwidth limitation, WorstBlock factor takes a maximum value allowing maximum number of mismatches to be replaced per frame.

As a result of applying these two control pointers, the plane is separated into four sections: A, B, C and D.

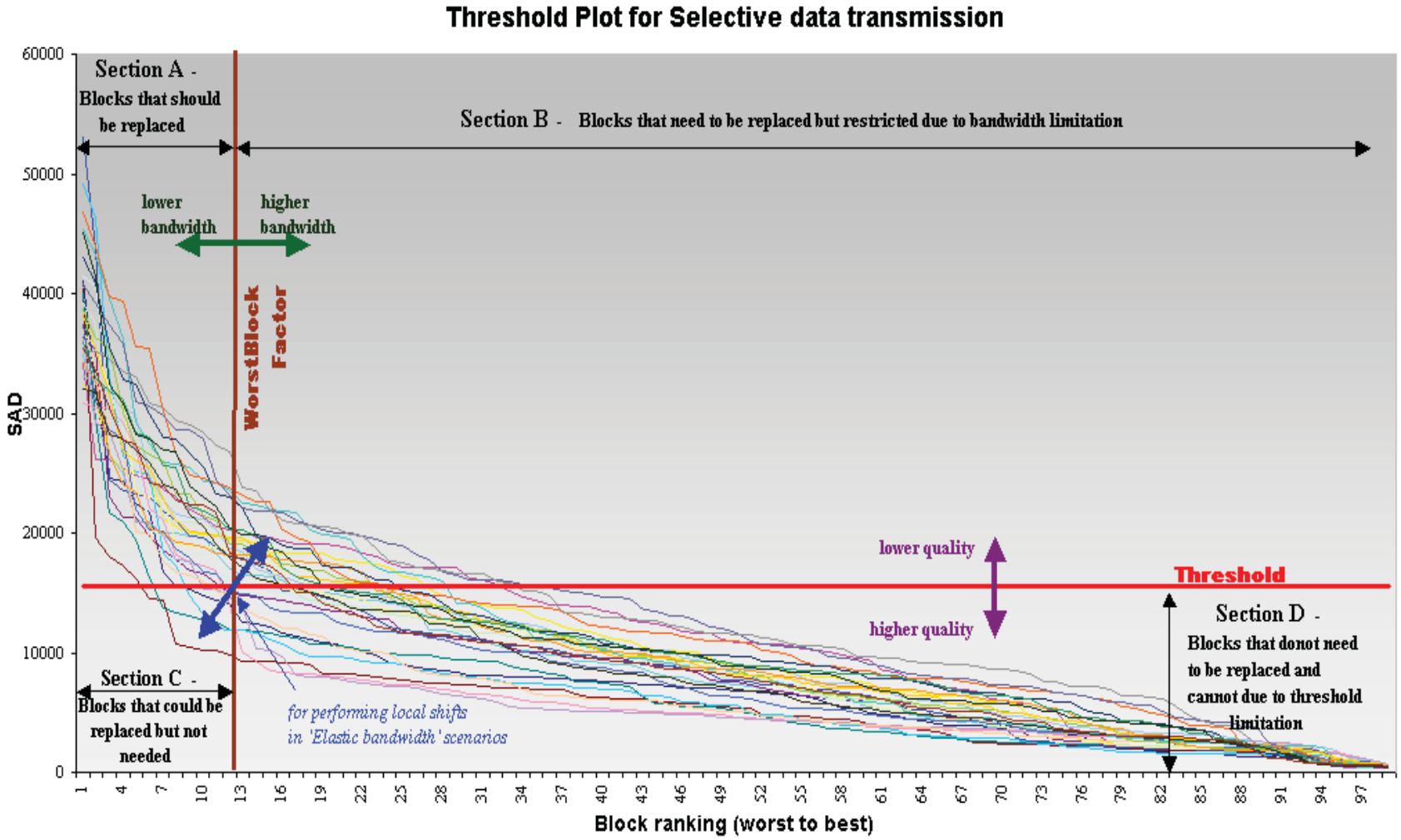


Figure 5.17: Threshold plot for selective data transmission.

(Each line represents a different frame)

Section **A**. This area represents the top-left corner of the plane containing the worst mismatches. Hence, all the blocks in this section are to be updated irrespective of the bandwidth availability. So, this section is referred to as ‘Replacement section’.

Section **B**. This is the top-right section and contains mismatched blocks as it is above the threshold level. The blocks in this section need to be replaced but the replacement can only be done if there is sufficient bandwidth. If bandwidth is constrained the blocks in this section are not corrected, but are counted only during higher bandwidth as this section merges with Section A.

Section **C**. The section below Section A, at bottom-left is C. Though blocks in this section can be replaced according to the available bandwidth, it is not necessary to do so as this will not result in a significant improvement in quality.

Section **D**. The majority of these blocks in the lower right exhibit stationary motion and therefore need not be replaced. Lowering the quality threshold and increasing the available bandwidth would result in more of these low-error blocks being updated.

Another feature worth mentioning, here, is the ability of this approach to handle local shifts in ‘Elastic bandwidth’ situations. It corresponds to the point of intersection of the Threshold and WorstBlock factor, indicated as the blue diagonal arrowed-line, and can be increased or decreased to accommodate local shifts. This feature is not included in our approach and saved for future work.

5.2.6 Error Correction

This section discusses about the way by which the above identified mismatch blocks are updated. The identified mismatches, or motion error Δ (x, y), are corrected either by (1) motion vector replacement or (2) block replacement.

Motion vector replacement

The mismatches occur as a result of translation error, which is introduced during multiresolution motion estimation. As the motion vector is flawed it would be appropriate to perform error correction by replacing the motion vector of the mismatched blocks. The new motion vector is obtained by repeating motion estimation but with reduced search area Ω′. The motion correction factor is defined by

$$\triangle(x,y) = arg \min_{i,j \epsilon \Omega'} \left\{ \sum_{i=1}^{N} \sum_{j=1}^{N} |I_t(x,y) - I_{t-1}(x+i, y+j)| \right\} \quad (5.9)$$

This motion correction is applied only for the identified mismatches while the others retain the same motion vectors.

Block replacement

This is a simple and a straightforward approach. In this method, the entire mismatch block is intracoded. This saves computation time and allows reusability of data available at the decoder. As the blocks are replaced from the same resolution, the motion exhibited by the block is free from error and it also improves the quality of the image.

The simulation result of the proposed resolution scalable framework with selective data transmission and the two error correction methods are presented in the next Chapter. It also discusses the performance of the scalable framework and evaluates the two error correction methods.