ABSTRACT

Interest in 3DTV has increased recently with more and more products and services becoming available for the consumer market. 3D video is an emerging trend in developing digital video system. Three-dimensional multi-view video is typically obtained from a set of synchronized cameras, which are capturing the same scene from different viewpoints.

The video (texture) plus depth (V+D) representation is an interesting method to realize 3D video. A depth map is simply a grayscale image which represents the distance between a pixel and camera in black and white. However, a major problem when dealing with multi-view video is the intrinsically large amount of data to be compressed decompressed and rendered.

We extend the standard H.264/MPEG-4 MVC for handling the compression of multi-view video. An algorithm is implemented to compress the data in which instead of separate bit-streams each for depth and texture, only one bit-stream for texture (also containing depth data) is developed. As opposed to the Multi-view Video Coding (MVC) standard that encodes only the multi-view texture data, the proposed algorithm performs the compression of both the texture and the depth multi-view sequences. The proposed extension is based on exploiting the correlation between the multiple camera views.

The goal of this thesis work is to establish an efficient method to encode depth information along with multiple but limited numbers of views

Software used is JMVC (Joint Multi-view Video Coding) jmvc8.0, which is an open source software for the Multi-view Video Coding (MVC) project of the Joint Video Team (JVT) of the ISO/IEC Moving Pictures Experts Group (MPEG) and the ITU-T Video Coding Experts Group (VCEG).

Chapter 1

Introduction

1.1 Motivation

Three-dimensional (3D) video and imaging technologies is an emerging trend in the development of digital video systems, as we presently witness the appearance of 3D displays, coding systems, and 3D camera setups. Three-dimensional multi-view video is typically obtained from a set of synchronized cameras, which are capturing the same scene from different viewpoints. This technique especially enables applications such as free-viewpoint video or 3D-TV.

A 3D experience such as for example in 3D-TV is obtained if the data representation and display enable to distinguish the relief of the scene, i.e., the depth within the scene. With 3D-TV, the depth of the scene can be perceived using a multi-view display that renders simultaneously several views of the same scene. To render these multiple views on a remote display, an efficient transmission, and thus compression of the multi-view video is necessary. However, a major problem when dealing with multi-view video is the intrinsically large amount of data to be compressed, decompressed and rendered.

A multi-view video system, can be explored in three different aspects. First, algorithms to acquire a depth signal from a multi-view setup. Second, efficient 3D rendering algorithms for a multi-view signal. Third, proposing coding techniques for 3D multi-view signals, based on the use of an explicit depth signal. This motivates that the scope of study is divided into three parts.

The first part addresses the problem of 3D multi-view video acquisition. Multi-view video acquisition refers to the task of estimating and recording a 3D geometric description of the scene. A 3D description of the scene can be represented by a so-called depth image.

The second part details the problem of multi-view image rendering. Multi-view image rendering refers to the process of generating synthetic images using multiple views.

We concentrate on the compression problem of multi-view texture and depth video. We extend the standard H.264/MPEG-4 MVC video compression algorithm for handling the compression of multi-view video. As opposed to the Multi-view Video Coding (MVC) standard that encodes only the multi-view texture data, the proposed encoder performs the compression of both the texture and the depth multi-view sequences. The proposed extension is based on exploiting the correlation between the multiple camera views.

The goal of a thesis is to establish an efficient method to encode depth information along with multiple but limited number of views.

1.2 Thesis Organization

To completely understand the problem of thesis a brief overview of the subject matters that are related to this research work is presented in chapter 2. Short discussions of the HD video coding standards along with a description of the advanced features of the H.264/AVC coder are also illustrated.

In chapter 3, an overview of the existing 3D video formats including both stereo and multi-view is provided. The merits of each format are discussed along with the drawbacks and limitations.

The entire process exercised to get the compressed data for multi view 3D application is discussed. An algorithm “Steganography” is implemented to compress the data in which instead of separate bit-streams each for depth and colour, only one bit-stream for colour (containing depth data) is developed. Complete methodology is described in the chapter 4.

In the chapter 5, the results we got from JMVC code which have been modified for implementing our algorithm of steganography is analysed. Also, comparison of existing 3D video formats with our new approach of steganography is discussed.

Finally, the future work in this area is proposed.

Chapter 2

Background

This chapter presents a brief overview of the subject matters that are related to this research work. After that, short discussions of the HD video coding standards along with a description of the advanced features of the H.264/AVC coder are illustrated.

2.1 Fundamentals of stereo visualization

Stereo visualization thus refers to the visual perception of the solid three-dimensional (3D) properties of some objects. Developments in stereo visualization were initiated in 1838 when Sir Charles Wheatstone described the "Phenomena of Binocular Vision" [1]. Binocular vision relates to the interpretation of two slightly different views of the same object seen by both human eyes. From two different views, Wheatstone showed that the viewer can mentally reconstruct the object in three dimensions.

To illustrate the concept, Wheatstone prototyped a device known as the "stereoscope", which paints two different images of the same object directly onto the viewer retina. The initial implementation of the stereoscope, illustrated by Figure 2.1, is composed of two mirrors, A and A', that project onto both eyes, the image of two different hand-drawn views E and E' of a wire-frame cube. It is interesting to note that these early developments in stereoscopic visualization were made prior to the invention of photography.

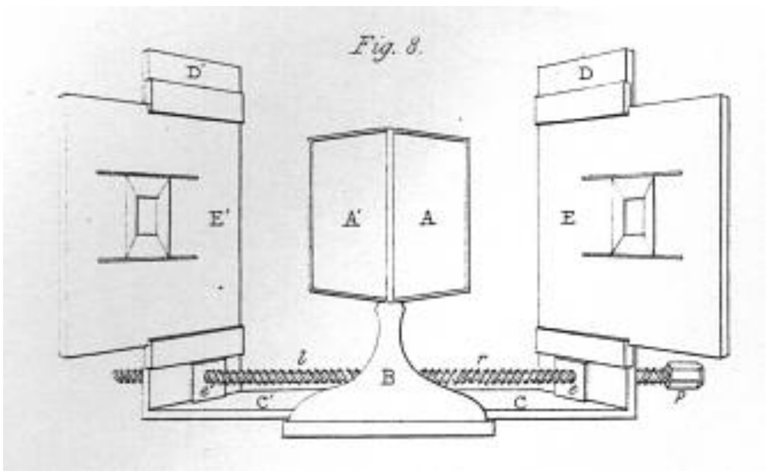


Figure 2.1: The initial implementation of the stereoscope: Based on two mirrors that project two views of the same object onto both human eyes. From these two views, the human visual system mentally derives a three-dimensional representation of the cube.

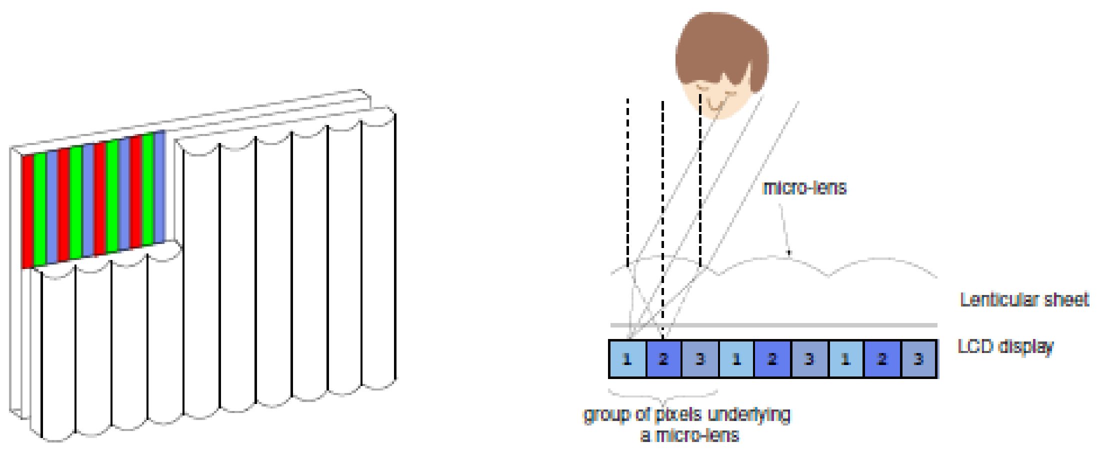
The presented conceptual principle that relates two 2D images to a 3D representation of an object, can be extended. More specifically, it can be intuitively deduced that a more accurate 3D description of the object can be obtained from a set a multiple views. Therefore, stereoscopic 3D properties of a scene can be derived from multiple views or multi-view images captured by a set of multiple cameras. For example, the background and foreground orientation and the relative positions of objects in the 3D scene can be extracted by analyzing the multi-view images. Starting with this elementary concept, we can now outline several applications for multi-view images.

2.2 Applications of multi-view imaging

2.2.1 Stereoscopic Displays

Stereoscopic displays allow the viewer to perceive the depth of the scene. This is achieved by displaying a left and right image/view in such a way that they are individually seen by the left and right eye. To obtain this result, several display technologies [2], including polarized displays, barrier stereo displays and lenticular displays have been developed. Stereoscopic lenticular displays or multi-view lenticular displays, are based on a lenticular sheet which is precisely positioned onto an LCD display see figure 2.2.

A lenticular sheet consists of an array of micro-lenses that directs the light of the underlying pixels in specific directions. Consequently, the viewing space in front of the display is divided into separate viewing zones, each of them showing a different image or view. The left and right eyes observe therefore a different view of the scene, thereby enabling the perception of depth. Figure 2.2 b, shows a three-view lenticular display that projects the light of three different pixels into three different viewing zones (two zones are drawn in the figure 2.2 b). To enable the viewer to watch the video scene from various viewpoints, nine-view lenticular displays have been introduced [3].



(a) (b)

Figure 2.2: Stereoscopic displays (a) A lenticular display: composed of a lenticular sheet precisely positioned onto an LCD display. (b) Multi-view lenticular display with three pixels/views covered by a micro-lens. Each view is projected into specific directions by the micro-lenses, so that the left and right eye see two different views.

It can be readily understood that increasing the number of views involves a loss of image resolution so that there exists a trade-off between the number of views supported by the display and the resolution of the image. Recently, the development of high-definition (quad

HD) LCD displays stimulated the usage of lenticular displays for various applications that we will be discussed in the sequel. On the short term, two-view displays for stereo-vision have gained strong popularity for 3D games and an early introduction to the 3D-TV market. Stereoscopic displays enable several 3D applications, which are briefly outlined below.

3D-TV for home entertainment : Three-dimensional television (3D-TV) is expected to become a key application of stereoscopic displays by providing the viewer a feeling of immersion in the movie.

Video games : Similarly, stereoscopic displays greatly enhance the realism of video games by showing a 3D representation of the virtual scene and characters.

Training and serious gaming systems : Another application for stereoscopic displays is the training of junior professionals. For example, it has been recently highlighted [4], that the usage of stereoscopic displays simplified the teaching of microscopical surgery to junior medical doctors. However, microscopes do not allow junior surgeons to perceive the depth as seen by the senior operating surgeon. This area can be extended to the emerging market of serious gaming that is between professional training and consumer gaming systems.

2.2.2 Free-viewpoint video :

The free-viewpoint video application provides the ability for users to interactively select a viewpoint of the video scene. This can be performed by capturing the video scene from multiple viewpoints. However, a freeviewpoint video system does not impose that the selected viewpoint corresponds to an existing camera viewpoint. Therefore, a free-viewpoint video application breaks the restriction of showing an event only from the viewpoint of the cameras, but instead, allows a free navigation within the 3D video scene. For example, interesting applications include the selection of an arbitrary viewpoint for visualizing and analyzing sports or dynamic art (e.g., dance) actions.

Sports. The ability to generate an arbitrary viewpoint is of particular interest for sports applications. For example, considering the case of a football match, it is often necessary for the referee to know the position of the players to ensure fair play. By rendering an appropriate viewpoint of the playing field, the player positions can be derived and illustrated using the virtual view.

Training video. Free-viewpoint video technologies also simplify video training activities. For example, the training of dynamic activities such as martial arts or dancing can be simplified by allowing the trainee to select a viewpoint of the scene.

2.2.3 Video editing and special effects :

Whereas in the previous subsection, the viewpoint was interactively chosen, in this subsection we summarize an application in which professional video editors manipulate time and place in one movie. For example, the multi-view image technology simplifies the production of special effects such as the "bullet time" effect. This effect provides the illusion to the viewer of freezing the time and gradually modifying the viewpoint of the scene. Such a special effect has been demonstrated in movies like "The Matrix". Additionally, by exploiting the 3D information, it is possible to discriminate some background or foreground objects, which is known as z-keying. Using 3D information, video objects can be easily removed and re-inserted into the video elsewhere. Such video editing capabilities were demonstrated for a dancing video sequence [5]. Finally, it is also possible to insert synthetic 3D objects in the scene to obtain an augmented-reality video scene. This allows a free composition of a virtual scene as desired by the director while preserving the photo-realism of the movie.

2.3 Video Coding Standards

Video compression is nothing but reducing the quantity of data and is a combination of spatial image compression and temporal motion compensation [6]. High degree of video compression became essential, due to the limitation of the transmission bandwidth. Most of the video compression techniques are lossy and degraded the reconstructed video quality compared to the original one after compression. Often this is because the compression scheme completely discards redundant information to achieve high compression efficiency. The available coding standards that are capable of compressing HD video are:

- H.264/ MPEG-4 Part 10 AVC [7, 8]
- MPEG-4 Part 2 [9]
- MPEG-2 [10, 11]
- Motion-JPEG2000 [12]

2.3.1 Coding Standard Comparison

MPEG-2 [10, 11] is a very popular and widely used video coding standard developed by Moving Pictures Expert Group (MPEG) and ITU-T, telecommunication standardization sector of International Telecommunication Union (ITU). MPEG-4 Part 2 [9] is an advanced video coding standard developed by MPEG and its documentation is first released in 1999. It includes many features of the MPEG-2 in addition to its advanced features. H.264/AVC is the newest video coding standard developed by a Joint Video Team (JVT) consisting of experts from the ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC Moving Picture Experts Group [7]. Motion-JPEG2000 [12] is a wavelet-based coding scheme developed by Joint Photographic Experts Group (JPEG) as an extension of the JPEG2000 [13] still image coding standard, and is an intra-frame only compression scheme, where each frame is compressed individually. Therefore, its compression efficiency is much less than inter-frame based coding standards, MPEG-2, MPEG-4 and H.264/AVC, where the redundancy between successive frames are considered to enhance compression efficiency. All these coding standards have the feature to support HD video resolution.

However, the H.264/AVC standard has a number of advantages over the MPEG-4 Part 2 and MPEG-2, which are summarized below:

• Bit rate saving: Compared to the MPEG-2 and MPEG-4 Part-2 codec, the H.264/AVC codec allows a higher compression ratio. For a given video quality, the H.264/AVC codec can achieve 39% and 64% of bit rate reduction compared to the MPEG-4 Part-2 and MPEG-2 codec respectively [14]. Compression efficiency of different coding standards are presented in Figure 2.3. According to [14], for the same video quality, if the MPEG-2 encoder requires 100 bits to encode a video, on average, the MPEG-4 Part 2 encoder will need 50.04 bits and the H.264/AVC encoder will need 36 bits to encode the same video.

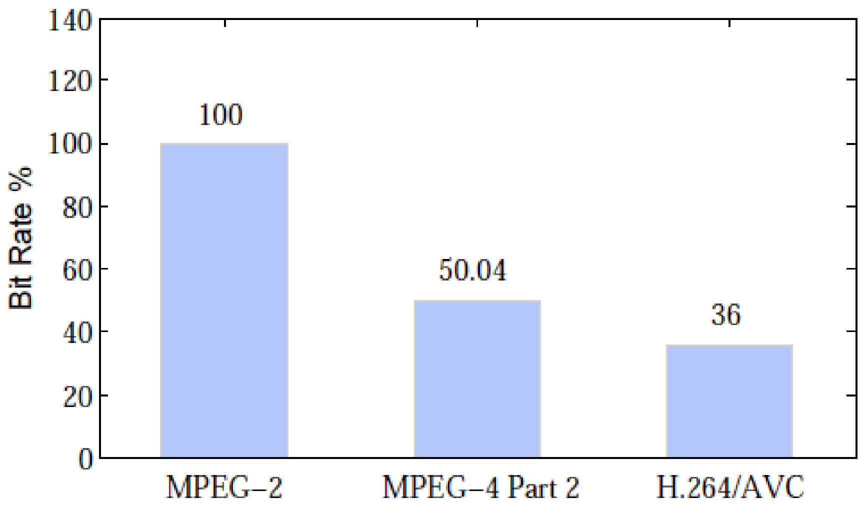


Figure 2.3: Compression efficiency between MPEG-2, MPEG-4, H.264/AVC

• High quality video: The H.264/AVC standard offers consistently high quality video at low and high bit rate.

• Error resilience: Compressed video streams are vulnerable to transmission errors, and that are almost inevitable in video transmission over wireless channels. The H.264/AVC standard provides advanced features to deal with the packet loss in packet networks and bit errors in error prone wireless networks [15, 16].

Considering the above advantages, especially the high compression efficiency, in this research work, the H.264/AVC standard is chosen as video compression tool.

2.3.2 H.264/MPEG-4 Part 10 AVC

The official name of the H.264/AVC is Advanced Video Coding (AVC) of MPEG-4 part 10 in ISO/IEC and H.264 in ITU-T, respectively [8]. The H.264/AVC encoder consists of two conceptual layers, the video coding layer (VCL)- defines the efficient representation of the video, and the network adaptation layer (NAL)- converts the VCL representation into a format suitable for specification transport layers or storage media [17]. Simplified block diagrams of the H.264/AVC encoder and decoder are presented in Figure 2.4 and Figure 2.5 respectively. Video coding layer of the encoder consists of three main functional units: prediction, core coding and entropy coding unit. The prediction unit includes a decoded video path to generate reference frames to enhance prediction efficiency. An H.264/AVC video decoder carries out the complementary processes of encoder to reconstruct the decoded video sequence.

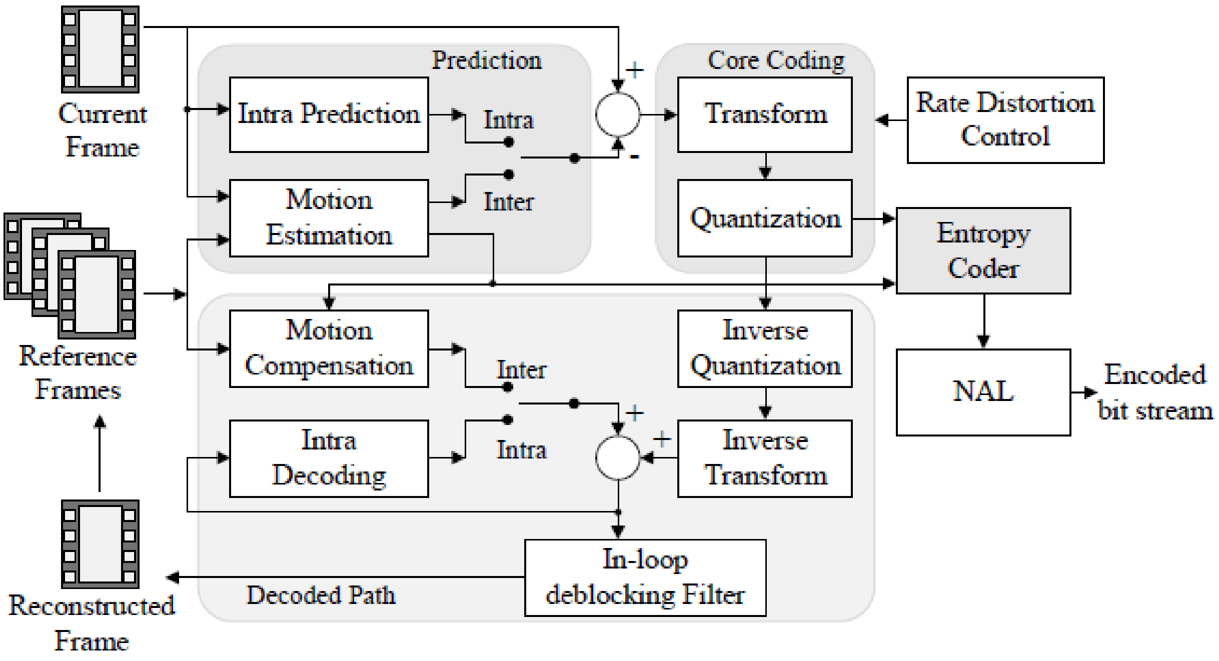


Figure 2.4: Block diagram of H.264/AVC encoder

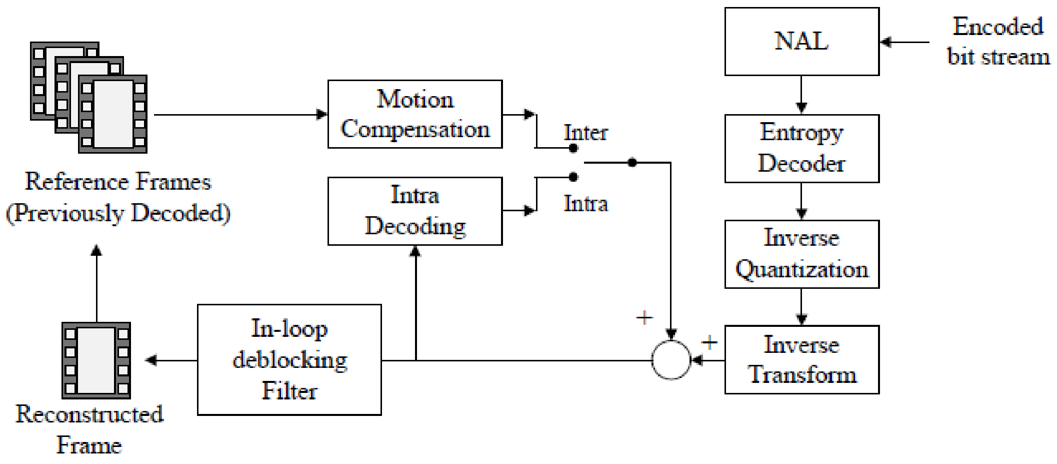
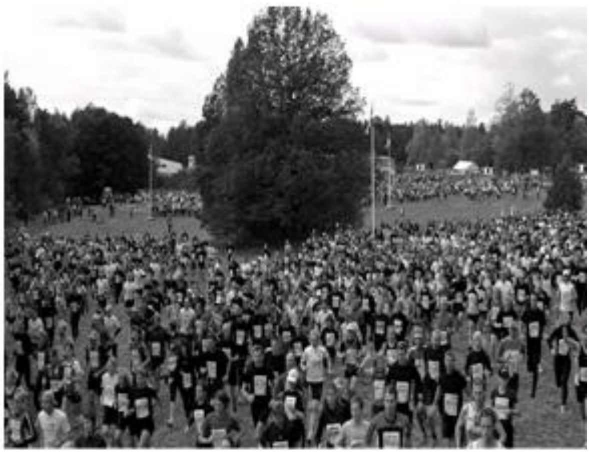


Figure 2.5: Block diagram of H.264/AVC decoder

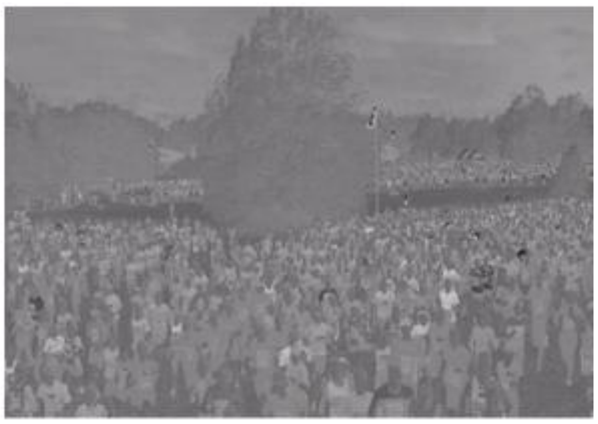
2.3.2.1 Prediction :

In video sequence, the data in pictures are often redundant in space and time. Therefore, prediction process is categorized in two sections: Intra prediction - considers the spatial redundancy in a video frame, and Inter prediction - take care of the temporal redundancy of the sequence. In H.264/AVC standard, each video frame is segmented into small blocks of pixel, called macroblocks (MB), and all the processing are performed on these macroblocks. Each macroblock consists of three components, Y, Cr, and Cb shown in figure 2.6. Y is the luminance component, which represents the brightness information of the image. Cr and Cb are the chrominance components, and represent the colour information of the image.

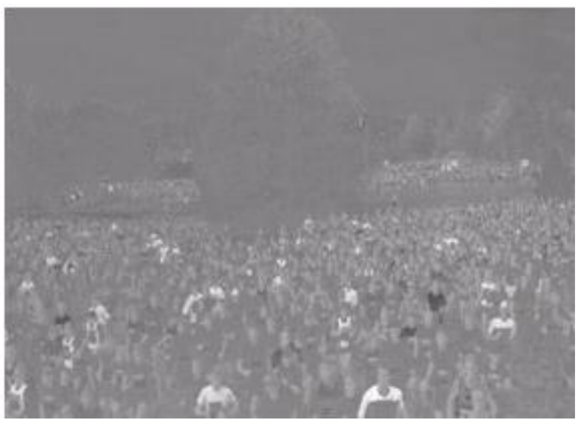
(a) Original video frame



(b) Y component



(c) U component



(d) V component

Figure 2.6: Luminance and chrominance components of an image

2.3.2.1.1 Intra Prediction

Intra prediction achieves moderate coding efficiency by exploiting correlation between adjacent macroblocks and eliminating spatial redundancy. In intra mode, an MB is predicted from neighbouring previously decoded and reconstructed macroblocks within the same frame (figure 2.7).

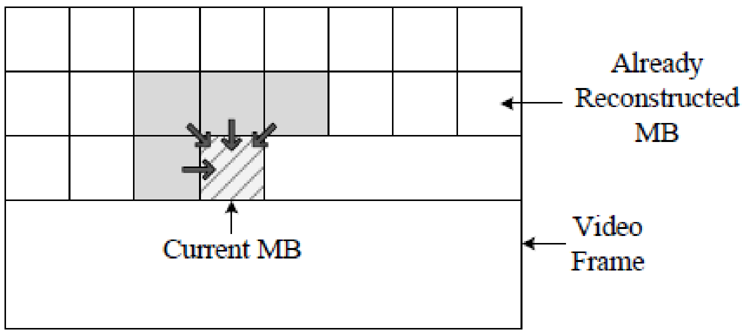


Figure 2.7: Intra Prediction

For the luminance (luma) component, intra prediction uses 16×16 and 4×4 macroblock sizes. There are a total of 9 optional prediction modes for each 4×4 luma block and 4 optional modes for a 16×16 luma block [18, 19]. Due to the less sensitiveness and importance of chrominance components in an image, each chroma block is down sampled by a factor of two in both vertical and horizontal directions. Therefore, if the luma prediction uses 16×16 MB, corresponding chroma block size will be 8×8 and will use a similar prediction technique. The frame, which contains only intra predicted macroblocks is called I-picture. It is an independent frame and used for predicting other frames.

2.3.2.1.2 Inter Prediction

Inter prediction takes the advantage of temporal correlation between successive frames of a video sequence to reduce the temporal redundancy. In inter prediction, the current frame is predicted from one or more previously encoded frames (figure 2.8). The H.264/AVC encoder uses block-based motion compensation and supports variable block-size motion compensation, ranges from 16×16 to 4 ×4 pixels [20]. In inter prediction method, to predict a macro-block in the current frame, at first, a search for closely matched MB in the previously decoded reference frames is conducted and the best matched MB is chosen. This process of finding the best matched MB is known as motion estimation (ME). The number of reference frame to be searched is defined by the parameter. The best matched MB is then subtracted from the MB of the current frame and the residue is known as motion compensation. The offset between the current block and the position of the candidate region is known as motion vector (MV). This motion compensation block, together with the motion vector are encoded and transmitted.

Different frames during inter prediction is presented in Figure 2.9 .The H.264/AVC encoder supports sub-pixel and quarter-pixel precision motion compensation [21]. There are generally two types of inter frames: P-picture and B-picture. P-picture is called forward predicted picture and is generated from previously encoded I or P pictures. It requires less number of coding data compared to I-pictures. B-picture is known as bi-directionally predicted picture and is reconstructed from earlier or later I or P pictures. Hence, it provides highest degree of data compression in comparison to I and P pictures.

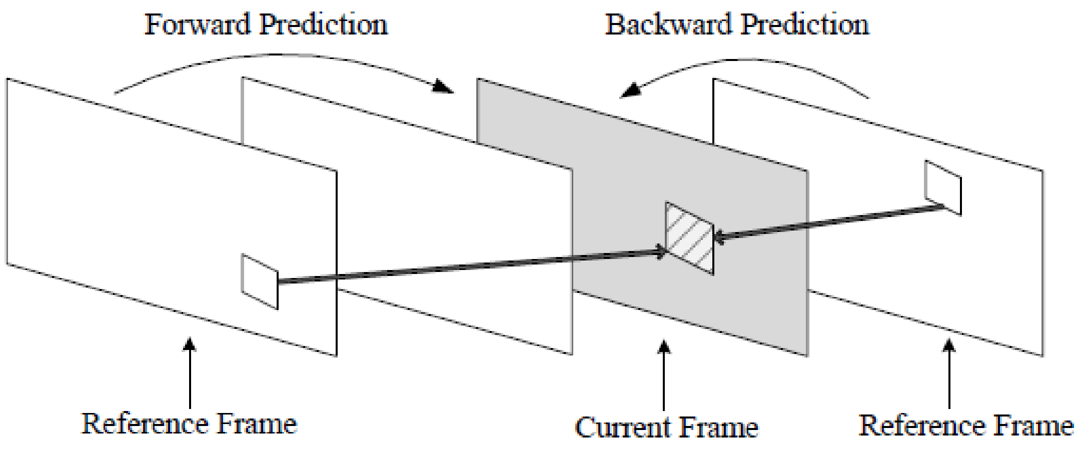
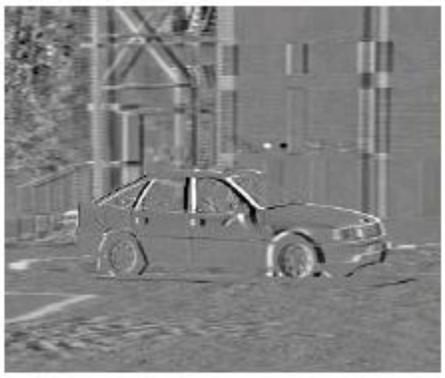


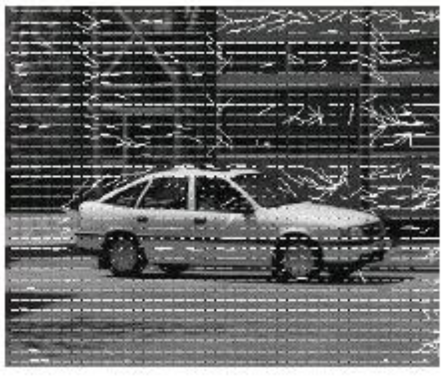
Figure 2.8: Inter prediction

(a) Frame 1

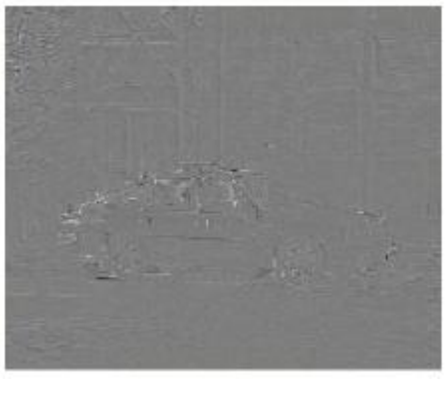
(b) Frame 2



(c) Residual (difference) frame



(d) Motion Vector



(e) Motion compensated residual frame

Figure 2.9: Frames during Inter prediction

2.3.2.2 Core Coding :

After the intra or inter prediction, the predicted MBs are transformed and quantized. In H.264/AVC encoder, an approximated 4×4 Discrete Cosine Transform (DCT), known as integer spatial transform is used. Here, the 4×4 approximated DCT is divided into two parts, 4×4 integer transform and fractional scalar multiplication factors that is further merged in quantization stage [22]. This small block base transform helps to reduce the blocking and ringing artifacts, and the integer transform eliminates the mismatch issues between the transform and inverse transform both in the encoder and decoder.

Significant portion of data compression is taken place in the quantization stage. However, quantization is a lossy process and introduces unavoidable and irretrievable quantization error which leads to degradation of decoded video quality. In H.264/AVC encoder, the quantization stage has 52 values of quantization parameters and corresponding quantization steps. The quantization step size increases by approximately 12% for increase in each quantization parameter [23]. This allows addressing a wide range of quality level in the decoded video in compensation to data compression. The fractional scaling factors mentioned above are incorporated into the quantization coefficients. After this stage, the quantized data are sent to entropy coding unit, as well as, to the feedback loop, to reconstruct the reference frames for inter prediction.

2.3.2.3 Entropy Coding :

Entropy coding is the last stage of video coding. It is a lossless coding technique that replaces the data element with coded representation, which reduces the data size significantly. In H.264/AVC, two modes of entropy coding - context adaptive variable length coding (CAVLC) [24] and context adaptive binary arithmetic coding (CABAC) [25] are used. The CABAC mode offers approximately 10%-15% enhanced compression efficiency compared to the CAVLC mode [25]. The CABAC consists of two stages. First, the blocks produced during the core coding process, transformed and quantized motion compensated macroblocks, motion vectors, and parameters, are converted into a sequence of binary symbols, known as binarization. Second, a binary arithmetic encoder is used to perform the compression.

2.3.2.4 In-Loop Deblocking Filter :

The H.264/AVC standard offers adaptive in-loop deblocking filter, which operates on the horizontal and vertical block edges within the motion-compensated prediction loop in order to remove the blocking artifacts and improve the quality of inter picture prediction [26]. In block oriented coding scheme, visually annoying blocking artifacts are introduced in various steps of encoding process, such as, during motion compensated prediction, in integer discrete cosine transformation of intra and inter predicted frames, due to coarse quantization of the transformed coefficients [27].

The aim of in-loop deblocking filter is to smooth the edges of each macroblock without affecting the sharpness of the picture, thus improving both objective and subjective quality of the decoded video. The filter is applied to the reconstructed frame both in encoder and decoder. After encoding HD video, the encoded bit stream is transmitted.

Chapter 3

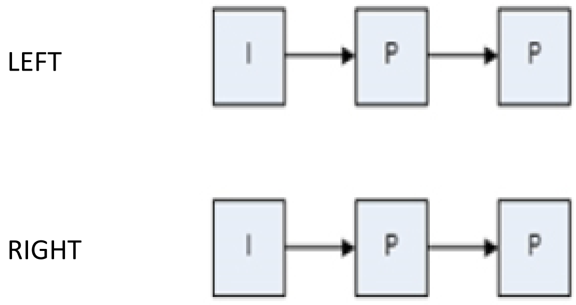
Literature Review

3.1 Existing 3D Video Formats

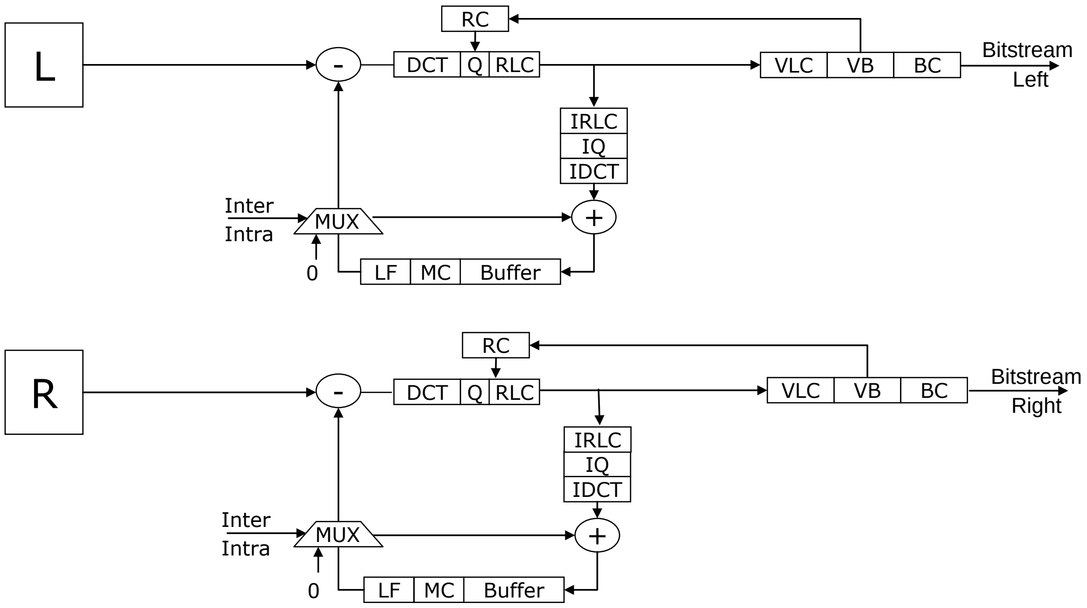
In this chapter, a brief overview of existing 3D video formats, including both stereo and multi-view formats, is provided. The merits of each format will be discussed along with the drawbacks and limitations.

3.1.1 Simulcast

The most obvious and straightforward means to represent stereo or multi-view video is simulcast, where each view is encoded independent of the other shown in figure 3.1. This solution has low complexity since dependencies between views are not exploited, thereby keeping computation and processing delay to a minimum. It is also a backward compatible solution since one of the views could be decoded for legacy 2D displays. With simulcast, each view is assumed to be encoded with full spatial resolution, and the main drawback is that coding efficiency is not maximized since redundancy between views is not considered. However, prior studies on asymmetrical coding of stereo, whereby one of the views is encoded with less quality, suggest that substantial savings in bitrate for the second view could be achieved. In this way, one of the views is more coarsely quantized than the other [28] or coded with a reduced spatial resolution [29], yielding an imperceptible impact on the stereo quality. Further study is needed to understand how this phenomenon extends to multiview video.



(a)



(b)

Figure 3.1: (a) H.264 Simulcast (b) Stereo encoder in H.264 multicast mode

3.1.2 Stereo Interleaving

There is a class of formats for stereo content that we collectively refer to as stereo interleaving. This category includes both time multiplexed and spatial multiplexed formats as shown in Figure 3.2. In the time multiplexed format, the left and right views would be interleaved as alternating frames or fields. With spatial multiplexing, the left and right views would appear in either a side-by-side or over/under format. As is often the case with spatial multiplexing, the respective views are “squeezed” in the horizontal dimension or vertical dimension to fit within the size of an original frame. The drawback of representing the stereo signal in this way is that spatial resolution would be lost.

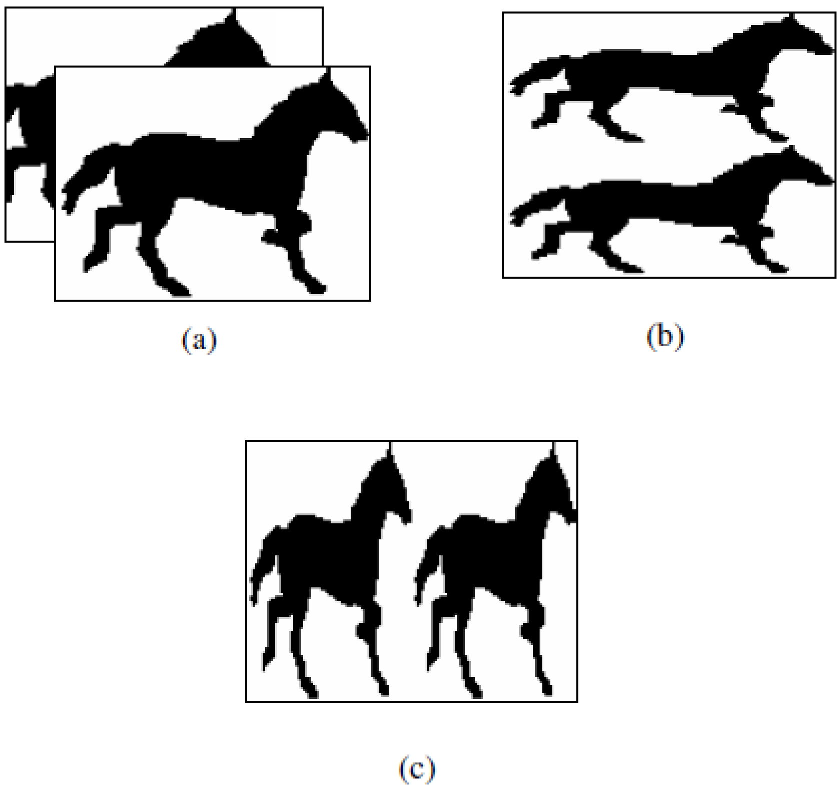


Figure 3.2: Various stereo interleaving formats (a) time multiplexed frames, (b) spatial multiplexed as over/under, (c) spatial multiplexed as side-by-side. Images from http://www.stereo3d.com.

To distinguish the left and right views, some additional out-of-band signalling is necessary. For instance, the H.264/AVC standard specifies a Stereo SEI message that identifies the left view and right view; it also has the capability of indicating whether the encoding of a particular view is self-contained, i.e., frame or field corresponding to the left view are only predicted from other frames or fields in the left view. Inter-view prediction for stereo is possible when the self-contained flag is disabled. Similar type of signaling would be needed for the spatially multiplexed content.

The major drawback of all stereo interleaving approaches is that it is not possible for legacy 2D devices to extract, decode and display a 2D version of the 3D program. These legacy devices are not designed to understand the special signalling for stereo content and will only be able to decode and display a stream with alternating frames or in the side by- side format. While this might not be so problematic for packaged media since special 3D disc players could be upgraded and new discs could store both 2D and 3D formats, it is certainly a major issue for broadcast services where the transmission channel is limited and devices cannot be upgraded.

3.1.3 2D + Depth

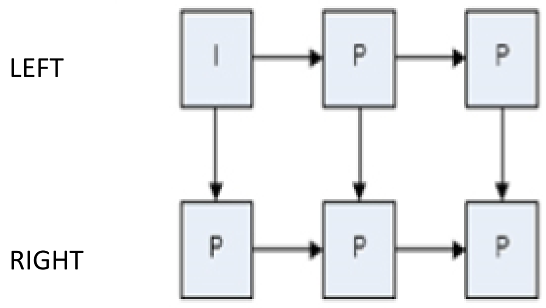
Another well-known representation format is the 2D plus depth format. The inclusion of depth enables a display independent solution for 3D that supports generation of an increased number of views as need by any stereoscopic display. A key advantage is that the main 2D video provides backward compatibility with legacy devices. Also, this representation is agnostic of coding format, i.e., the approach works with both MPEG-2 and H.264/AVC. The main drawback is that the format is only capable of rendering a limited depth range and has problems with occlusions.

ISO/IEC 23002-3 (also referred to as MPEG-C Part 3) specifies the representation of auxiliary video and supplemental information. In particular, it enables signalling for depth map streams to support 3D video applications.

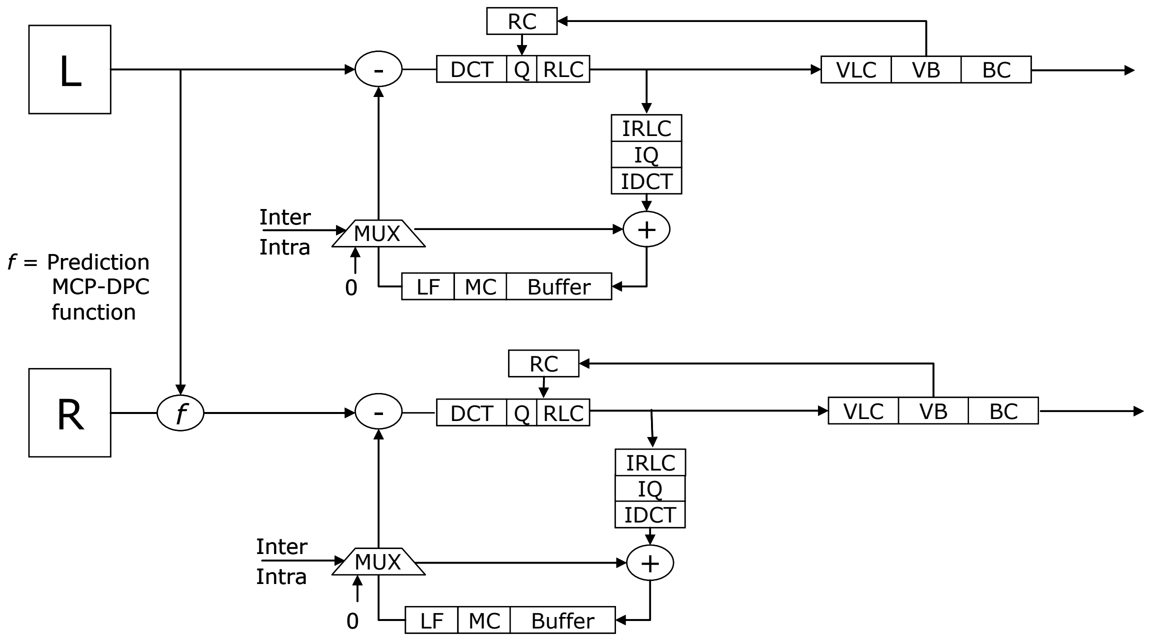
3.1.4 Multiview Video Coding

To improve coding efficiency of multiview video, the redundancy over time and across views could be exploited. In this way, pictures are not only predicted from temporal neighbors, but also from spatial neighbors in adjacent views. The Multiview Video Coding (MVC) standard has been specified as an amendment of H.264/AVC. It has been shown that MVC gives significantly better results compared to simple AVC-based simulcast [30]. Specifically, improvements of more than 2 dB have been reported for the same bitrate, and subjective testing has indicated that the same quality could be achieved with approximately half the bit-rate for a number of test sequences. A key aspect of the MVC design is that it does not require any changes to lower-level syntax, so it is quite compatible with single-layer AVC hardware. Also, it is mandatory for the compressed multiview stream to include a base layer stream that could be easily extracted and used for backward compatibility with legacy 2D displays; this base layer stream is identified by the NAL unit type in H.264/AVC.

Furthermore, inter-view prediction is enabled through flexible reference picture management, where decoded pictures from other views are made available in the decoded reference picture buffer. It is important to emphasize that the core decoding modules do not need to be aware of whether reference picture is a time reference or multiview reference from another view. In terms of syntax, the standard only requires small changes to high-level syntax, e.g., view dependency needs to be known for decoding. MVC was designed mainly to support auto-stereoscopic displays that require a large number of views. However, large camera arrays are not common in the current acquisition and production environments. Furthermore, although MVC is more efficient than simulcast, the rate of MVC encoded video is still proportional to the number of views. Of course, this varies with factors such as scene complexity, resolution and camera arrangement, but when considering a high number of views, the achievable rate reduction might not be significant enough to overcome constraints on channel bandwidth.



(a)



(b)

Figure 3.3: (a) MVC (b) Stereo Encoder in MVC mode

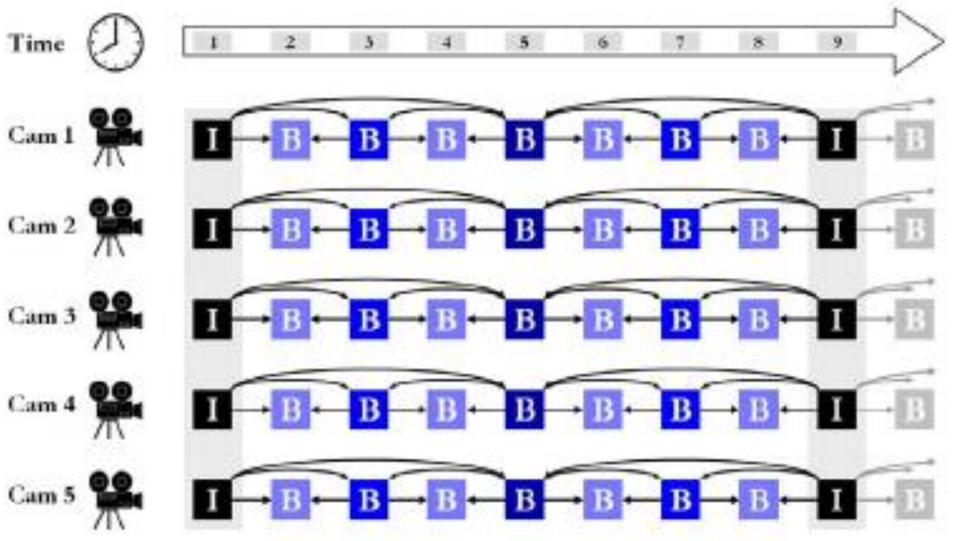


Figure 3.4: Simulcast coding structure with hierarchical B pictures for temporal prediction (black arrows).

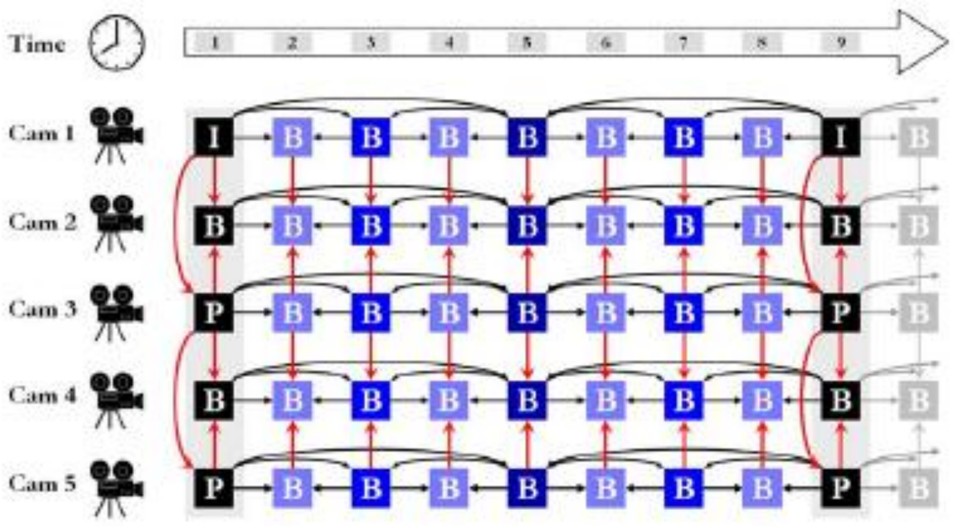


Figure 3.5: Multi-view coding structure with hierarchical B pictures for both temporal and inter-view prediction (red arrows).

It is important to note that in the near-term MVC is still useful for delivery of stereo contents. While some rate reduction could certainly be achieved relative to simulcast, the more important factor might be the backward compatibility that it provides to existing 2D services. In contrast to some of the stereo interleaving solutions, such as side-by-side, the full resolution could also be maintained. MVC also has benefits relative to the 2D + Depth format for stereo in terms of rendering quality for generic scenes.

Chapter 4

Methodology Adopted

In this chapter, the entire process exercised to get the compressed data for multi view 3D application is discussed. An algorithm is proposed to compress the data in which instead of separate bit-streams each for depth and colour, only one bit-stream for colour (containing depth data) is developed.

4.1 Introduction

A major problem when dealing with multi-view video is the intrinsically large amount of data to be compressed, decompressed and rendered. Therefore, coding algorithms enabling an efficient compression of both multi-view depth and texture video are necessary.

In the existing 3D Video formats large amount of data is compressed and the channel bandwidth is directly proportional to number of views and depth maps as with the MVC.

4.2 Steganography

In the earlier implementations, channel bandwidth is directly proportional to number of views and depth maps as with MVC, but our objective is to have a single compressed bit stream of multiple views along-with depth maps (A depth map is simply a grayscale image which represent the distance between a pixel and its left counterpart using a gray-scale value between black and white) which consumes lesser channel bandwidth.

So to get single bit stream having information of both colour and depth, we implement the concept of Steganography. The word **steganography** literally means covered/protected writing as derived from Greek. Steganography is the art and science of writing hidden messages in such a way that no one, apart from the sender and intended recipient, suspects the existence of the message, of hiding depth map along-with the colour for corresponding views (steganography). In this method single bit stream is generated, in which depth information is hidden in the information of colour encode bits.

4.3 Encoding phase

4.3.1 For Depth Sequence:

The idea is that, we need to have depth encoded bit stream (MESSAGE) for each frame so that we can merge this message into the colour encoded bits for the respective frame. So to get the encoded Depth Bits, each frame of Depth is predicted, transformed quantized and encoded (entropy encoding) as usual obtaining a sequence of bit that we will call MESSAGE.

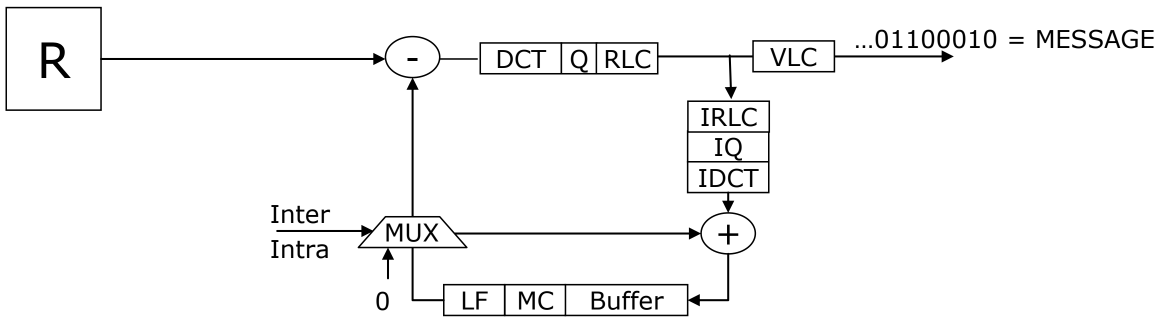


Figure 4.1: Encoding Depth frame whose bits are to be merged at the later stage.

4.3.2 For Colour Sequence:

Now before hiding the information of depth map frame into the colour frame, firstly, for each frame of the colour, 4x4 blocks are extracted, transformed, quantized and encoded using run-length coding obtaining a sequence as follows:

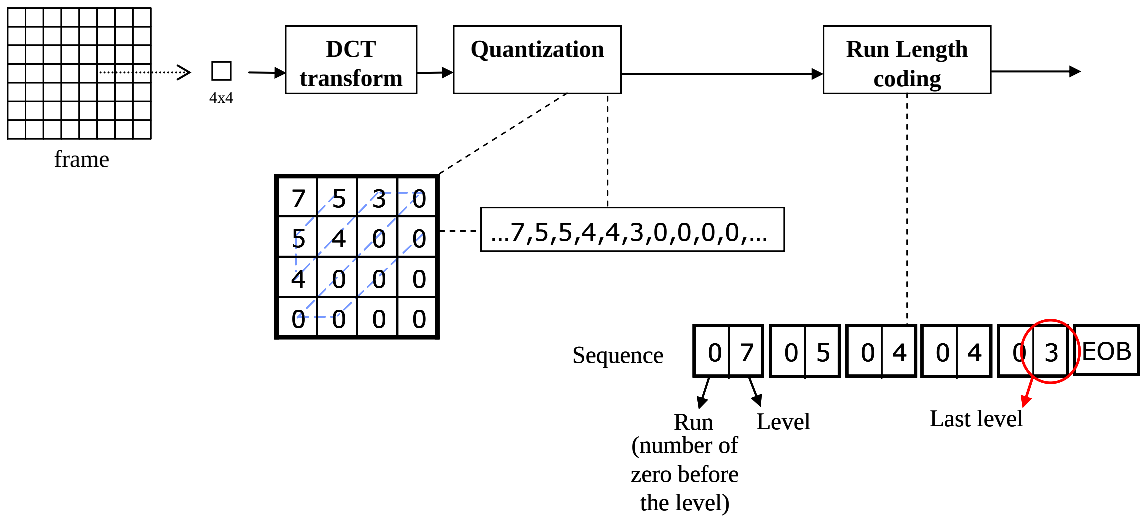
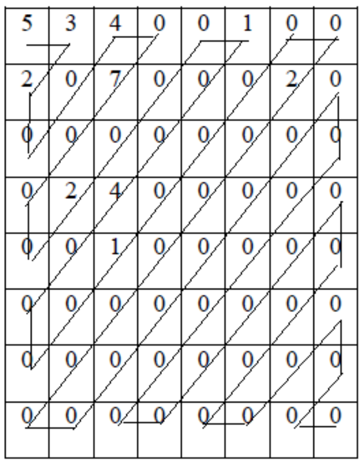


Figure 4.2: Encoding of Colour frame till RLC before merging.

Algorithm should closely follow the Compression standard to avoid computationally demanding operations, like DCT and IDCT transform or motion vector calculation. Therefore, the algorithm should work on the lowest layer, called the block layer, in which spatial 8x8 blocks are represented by 64 quantized DCT-coefficients. Figure 4.3 shows the different domains in which such a quantized DCT-block can be represented.

The first domain in the block-layer is the coefficient domain *(cd).* In this domain a block contains N*N (N=8) integer entries that correspond with the quantized DCT-coefficients. Many of the entries are usually zero, especially those entries that correspond with the spatial high frequencies. In the run-level domain, the non-zero AC coefficients are re-ordered in a zig-zag scan fashion and are subsequently represented by a tuple *(r,l)*, where the run *(r)*

N*N Blocks | tuples *(r,l)* | VLC Codewords



(0,5), (0,3), (0,2), (2,4),

(1,7), (3,2), (3,1), (2,4),

(4,1), (5,2)

001001100
001010
01000
0000000101000
00000010100
001001000
001110
0000000101000
001100
00000011110
10

Coefficient Domain *(cd)* | run-level domain | bit domain *(bd)*

Figure 4.3: DCT-block representation domain.

is equal to the number of zeros preceding a certain coefficient and the level *(l)* is equal to the value of the coefficient. In lowest level domain, the bit domain *(bd),* the *(r,l)* tuples are represented by variable length coded (VLC) codewords. The codewords for a single DCT-block are terminated by an end of block (EOB) marker.

4.3.3 Merging

The Depth bitstream (say Message) L, consisting of bits L_i (i= 0,1,2,…, *l*) is embedded in the colour bitstream by forcing the LSB (least significant bit) of their quantized last level to the value of L_i.

Each least significant bit (LSB) of the last level of the sequence is replaced with a single message bit. In this way data is hidden in "noise" of image. Change in the LSB of the last level of the sequence obtained after run-length coding introduces only a little amount of noise which is almost invisible to sight.

The DC coefficients in intra blocks are not used, because they are predicted from other DC coefficients and are very significant in terms of containing valuable information.

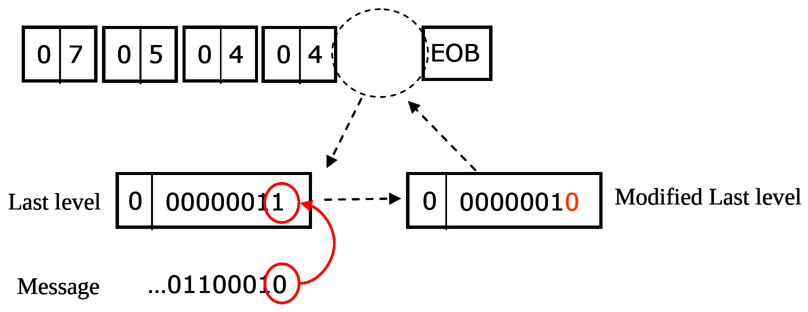


Figure 4.4: Procedure showing the merging concept

To add the depth bitstream to a colour bitstream, the DCT coefficients in each block are tested according to the scan order, till we get the last non-zero coefficient. If a last non-zero coefficient is found, and the least significant bit of its level is unequal to the depth bit L_i (I = 0,1,2,…, l), this LSB is replaced by the depth bit L_i . And if the LSB of its level equals the depth bit L_i the LSB is not changed. The procedure is repeated until all Depth bits are merged/ embedded.

Now the modified sequence is encoded as usual using entropy encoding (VLC).

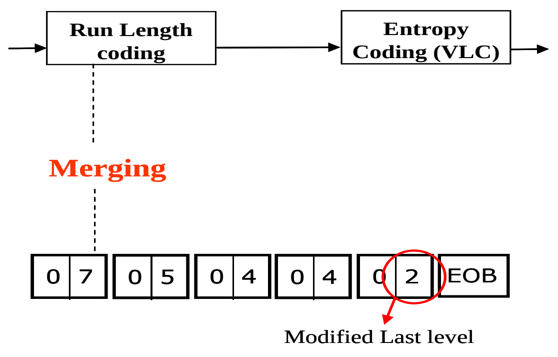


Figure 4.5: Modified last level, ready for Entropy coding

Repeat the procedure for each block until all bits of the message are inserted into the colour bitstream.

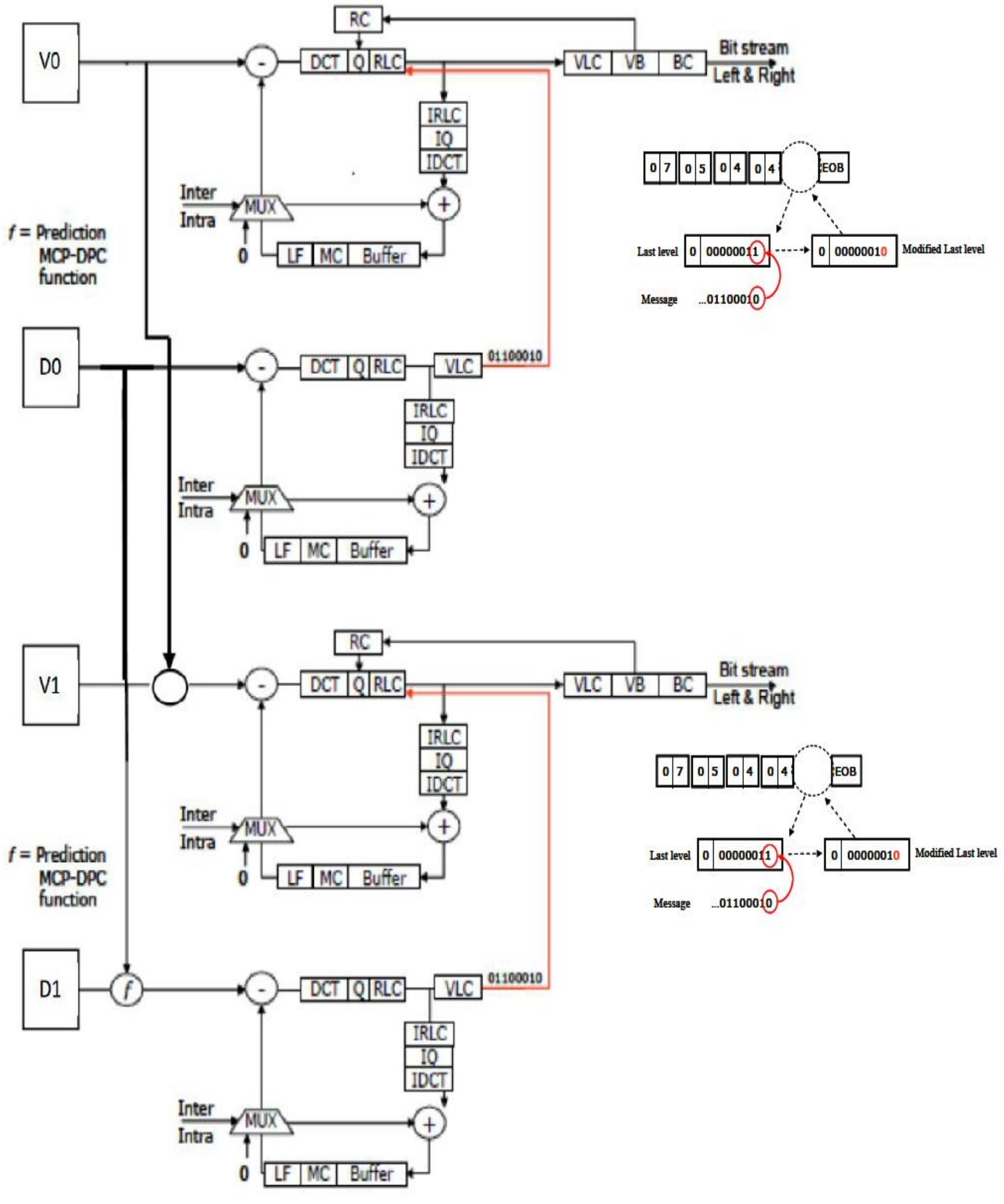


Figure 4.6: Steganography approach in Multi-View: D0 merged into V0, D1 predicted from D0 then merged into V1 which may be predicted from V, And so on....

4.4 Decoding phase

4.4.1 For Depth Sequence

Inverse VLC is performed obtaining the correspondent of the encoded sequence.

Extraction: the least significant bit of each last level of the sequence is identified and collected to reconstruct the message.

Decoding continues as usual.

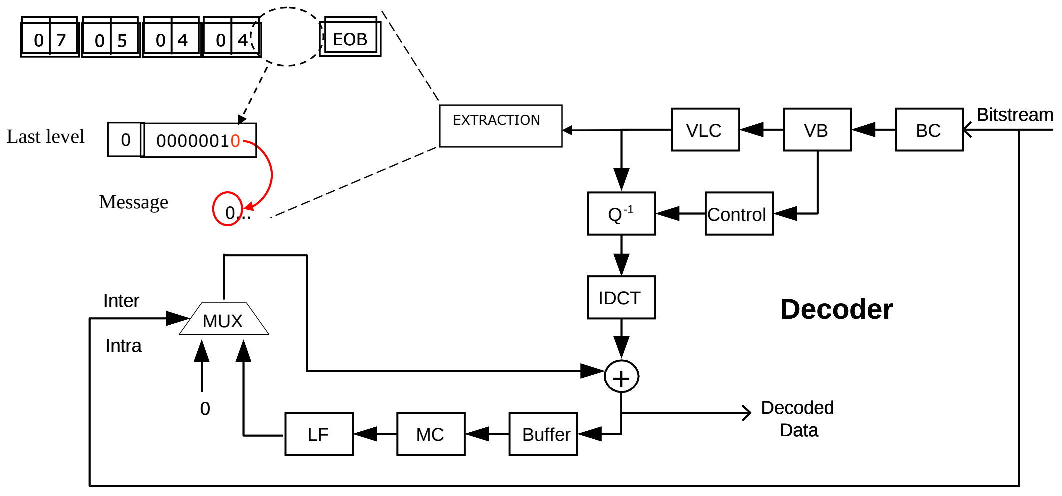


Figure 4.7: Extraction at decoder side

4.4.2 For Colour Sequence

Inverse VLC is performed on the message reconstructed from Depth and decoding continues as usual.

4.5 Software Used

JMVC (Joint Multi-view Video Coding) jmvc8.0, software is an open source software for the Multi-view Video Coding (MVC) project of the Joint Video Team (JVT) [31] of the ISO/IEC

Moving Pictures Experts Group (MPEG) and the ITU-T Video Coding Experts Group (VCEG).

This is basically used to encode & decode only the colour (views) of the video frames as the input but not the depth map as input. So some modifications had been done to encode the depth map also.

4.6 Implementation of Previous concepts in JMVC

Using the same JMVC with modifications, different forms of MVC implementation have been done as:

1. Multiple colour views in a single stream and Multiple Depth views in another stream. So getting 2 separate bit streams each for colour and depth, each of the bit streams have data corresponding to n views, if n is the number of multiple views.

2. Another implementation, we have single bit stream file having both colour and depth data. So, depth map is encoded as a separate view but inter-view predictions are among the views. Similarly, inter-view predictions are among the 'depth-views'.

But in these implementations, channel bandwidth is directly proportional to number of views and depth maps as with MVC. These implementations are useful for benchmarking purposes.

Objective is to have a single compressed bit stream of multiple views along-with depth maps which consumes lesser channel bandwidth.

4.7 Implementation done for Steganography

For the implementation of this idea, we need to have depth encoded bit stream (MESSAGE) for each frame so that we can merge this message into the colour encoded bits for the respective frame.

So to implement this concept, we have done these **sub-tasks** as:

1. ***Encoding of depth-map only***, using the same JMVC code but after some modifications. In this we are encoding only the luma component and not the chroma components of the video frame.

2. *Encoding of colour (view) and depth map* have been done together. In one pass depth-map is encoded for a frame and in second pass colour view is encoded for the corresponding frame, so two passes for one frame, and this continues to go on for other frames also. We are also getting two different reconstructed videos (recon_colour_Y.yuv, recon_depth_Y.yuv) one for colour and another for depth.

3. *Dumping*: Now to merge the encoded depth frame bits in colour frame bits we have stored these depth bits in to a file (e.g. dump_0.bin) for a depth frame 0. This will be used in second pass while encoding corresponding view (colour frame). dump_X.bin (X being 0, 1, 2 etc, the frame number in encoding order) contains data only corresponding to the slice_data () without any parameters (SPS, PPS, etc) and without slice header. This is assuming that the picture of depth map uses the same parameters, except the slice_data(), as that of the corresponding picture of view (colour).

4. *Merging*: The data which has been dumped in dump_X.bin is merged with the encoded bits of colour at the macro-block level. Some points are taken care while doing merging

- Merging is done in each transform block except in the blocks of I_16x16_DC and Chroma_DC residual mode. The DC coefficients in intra blocks are not used, because they are predicted from other DC coefficients and are very significant in terms of containing valuable information.

- Also for seamless decoding the bitstream (merged depth in colour), merging is avoided when
  - If uiNumSigCoeff = 0 i.e. Number of significant coefficients are zero. uiNumSigCoeff is nonzero when there is atleast one non-zero coefficient in the 4x4 block.
  - If the uiNumSigCoeff $\neq 0$, but after merging the last level coefficient value of a block is changing from "1" to "0". Merging is avoided here to keep uiNumSigCoeff unchanged.

The bitstream we get after merging is processed as usual through VLC (variable length coding) i.e. entropy coding. After the entropy coding the final encoded bit stream obtained is consistent with 2D decoder.

Chapter 5

Results and Analysis

In this chapter we will discuss the results we got from JMVC code which we have modified for implementing our algorithm of steganography. Also, comparison of existing 3D video formats with our new approach of steganography is discussed.

For evaluating our algorithm, we have taken four sequences i.e. MSR_BreakDancer, MSR_Ballet, Nokia_Dancer, Kendo. These four sequences are given as an input to our code and compiled at different sets of QP (quantization parameters).

For benchmarking purpose we have also used the earlier implemented code in which one bit-stream is generated but take texture and depth as different views.

5.1 Using Same Quantization Parameter

Sequence used here is MSR_BreakDancer, in which both Texture and Depth are encoded at Quantization parameter (QP) = 28, Group of Pictures (GOP) = 8, Intra period = 16, Number of frames = 15 (I B B B B B B B P B B B P B P).

Table 1 is shown below in which “bits to be merge” are the number of encoded depth data bits which are to be merged in colour frame. Bits merged are bits merged after steganography.

Table 1 : Merge Statistics for MSR_BreakDancer sequence with same QP for Depth & Texture

Encoding Order	POC	Picture Type	Depth(QP=28) & Colour(QP=28)		
			Bits to be Merge	**Bits Merged**	**Bits Left**
0	0	I	13568	1446	12122
1	8	P	14504	973	13531
2	4	B	7160	311	6849
3	2	B	5656	233	5423
4	6	B	5496	198	5298
5	1	B	4304	142	4162
6	3	B	3488	150	3338
7	5	B	3752	119	3633
8	7	B	4024	112	3912
9	12	P	8688	469	8219
10	10	B	6296	285	6011
11	14	P	5696	339	5357
12	9	B	4112	126	3986
13	11	B	4816	176	4640
14	13	B	4672	170	4502

Here merging of bits are too low due to

- Skipping of Macro-blocks: around **55%** of macro blocks are skipped in this sequence using the above configuration. Because of which very large number of merging is skipped.

- Skipping blocks (residuals): In **8%** of total blocks in a frame the merging is skipped, the reasons for this is the invalid Cbp i.e. coded block pattern. Cbp becomes invalid when there is no non-zero coefficient present in the block. Skipping takes place in Luma as well as in Chroma blocks due to

    - Coded_Block_Pattern_Luma

    - Coded_Block_Pattern_Chroma

Table 2 : Statistics of skipping of Macroblocks for MSR_BreakDancer Sequence

Encoding Order	POC	Picture Type	MB Skip	
			Macro Blocks	% frame size(MB)
0	0	I	0	
1	8	P	148	37.37
2	4	B	165	41.67
3	2	B	189	47.73
4	6	B	193	48.74
5	1	B	250	63.13
6	3	B	240	60.61
7	5	B	254	64.14
8	7	B	264	66.67
9	12	P	168	42.42
10	10	B	193	48.74
11	14	P	215	54.29
12	9	B	258	65.15
13	11	B	221	55.81
14	13	B	246	62.12
			3004	**54.18**

Table 3 : Statistics of skipping of Blocks due to invalid Cbp_luma for MSR_BreakDancer Sequence

Encoding Order	POC	Picture Type	Blocks skipped due to Luma_Scan Due to Cbp	
			Blocks	% of Frame size (Blocks)
0	0	I	257	4.06
1	8	P	360	5.68
2	4	B	687	10.84
3	2	B	661	10.43
4	6	B	665	10.50
5	1	B	475	7.50
6	3	B	527	8.32
7	5	B	480	7.58
8	7	B	447	7.05
9	12	P	470	7.42
10	10	B	611	9.64
11	14	P	460	7.26
12	9	B	466	7.35
13	11	B	576	9.09
14	13	B	487	7.69
			7629	**8.03**

- Skipping of merging also takes place when Significant coefficients i.e. uiNumSigCoeff = 0. uiNumSigCoeff is nonzero when there is at least one non-zero coefficient in the 4x4 block. Approximately in **7%** of the total blocks in this sequence merging is skipped.

Table 4 : Statistics of skipping of blocks due to uiNumSig = 0, for MSR_BreakDancer Sequence

Encoding Order	**POC**	**Picture Type**	**uiNumSig = 0**	
			Blocks	% of Frame size (Blocks)
0	0	I	2315	36.54
1	8	P	1212	19.13
2	4	B	401	6.33
3	2	B	247	3.90
4	6	B	227	3.58
5	1	B	186	2.94
6	3	B	166	2.62
7	5	B	127	2.00
8	7	B	122	1.93
9	12	P	608	9.60
10	10	B	300	4.73
11	14	P	392	6.19
12	9	B	134	2.11
13	11	B	220	3.47
14	13	B	200	3.16
			6857	**7.21**

- Skipping of merging also takes place, if the uiNumSigCoeff ≠ 0, but after merging the last level coefficient value of a block is changing from “1” to “0”. Merging is avoided here to keep uiNumSigCoeff unchanged. Approximately in **2%** of the total blocks, merging is skipped due to this reason.

Table 5 : Statistics of skipping of blocks due to changing of last level from 1 to 0, for MSR_BreakDancer Sequence.

Encoding Order	POC	Picture Type	Last Level 1 to 0	
			Blocks	% of Frame size (Blocks)
0	0	I	635	10.02
1	8	P	431	6.80
2	4	B	164	2.59
3	2	B	76	1.20
4	6	B	75	1.18
5	1	B	76	1.20
6	3	B	56	0.88
7	5	B	58	0.92
8	7	B	42	0.66
9	12	P	219	3.46
10	10	B	115	1.82
11	14	P	101	1.59
12	9	B	52	0.82
13	11	B	68	1.07
14	13	B	66	1.04
			2234	**2.35**

5.2 Using different Quantization Parameter

So, the other way of doing this is to encode the depth at higher QP than that of texture. Because by doing so the number of encoded bits for depth frame becomes less, and the number of bits for the texture frame remains the same. So, the bits to be merged become less as compared to the previous method of encoding at same QP.

Here we are taking four different above mentioned sequences, and encoding each of them at different sets of QP. Depth is encoded at QP $_{color}$ +18, stated as "Steganography".

For the benchmarking purpose, each sequence is also encoded by the earlier implemented code in which one bit-stream is generated but take texture and depth as different views stated as "Depth as Views".

5.2.1 Sequence : MSR_BreakDancer

Table 6 : "Steganography" Results for MSR_BreakDancer sequence

QP		BITRATE(kbits/sec)	PSNR(Y) Colour (dB)
14 , 32	View 0	3144.96	38.8612
	View 1	2691.12	39.0253
	Overall	**5836.08**	**38.94325**
18 , 36	View 0	1780.3333	33.3834
	View 1	1436.5867	33.2114
	Overall	**3216.92**	**33.2974**
22 , 40	View 0	1026.7733	31.9291
	View 1	794.4667	32.402
	Overall	**1821.24**	**32.16555**
26 , 44	View 0	606.5733	31.3128
	View 1	451.3333	31.271
	Overall	**1057.91**	**31.2919**

Table 7 : "Depth as Views" Results for MSR_BreakDancer sequence

QP		BITRATE(kbits/sec)	PSNR(Y) (dB)
14	View 0 C	3019.4933	48.8091
32	View 1 D	164.24	40.8067
14	View 2 C	2579.28	48.7566
32	View 3 D	109.16	41.1837
	Overall	**5872.17**	**44.889025**
18	View 0 C	1706.6133	46.0777
36	View 1 D	98.2267	38.1138
18	View 2 C	1384.6667	46.1211
36	View 3 D	65.0267	38.4278
	Overall	**3254.53**	**42.1851**
22	View 0 C	985.0667	43.7127
40	View 1 D	56.8533	35.5108
22	View 2 C	765.5867	43.8143
40	View 3 D	40.4	35.9166
	Overall	**1847.91**	**39.7386**
26	View 0 C	582.64	41.2682
44	View 1 D	34.52	33.175
26	View 2 C	432.84	41.3139
44	View 3 D	26.4133	33.2809
	Overall	**1076.41**	**37.2595**

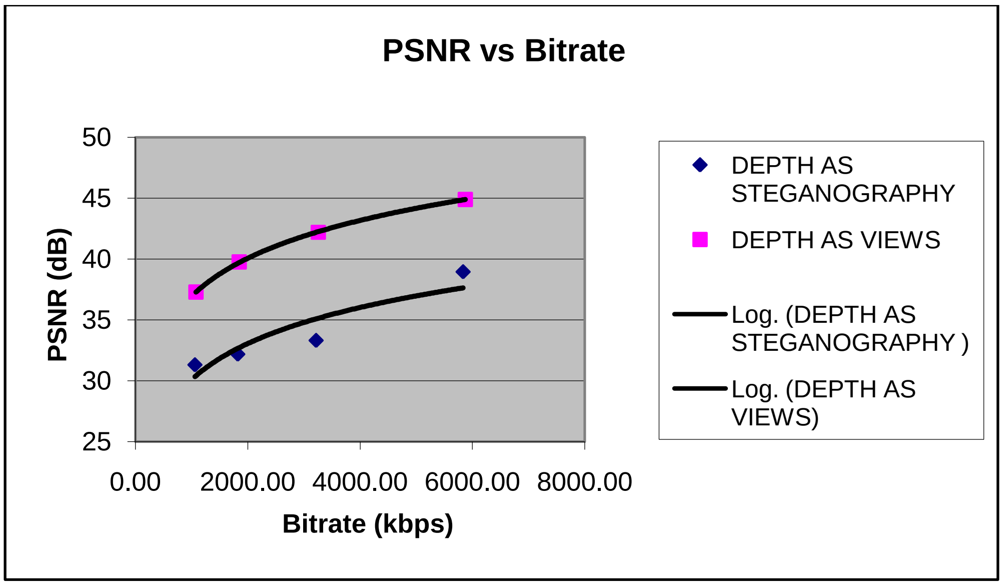


Figure 5.1 : Bitrate vs PSNR graph for MSR_BreakDancer sequence

Table 8 : Merge Statistics for MSR_BreakDancer sequence with same QP for Depth & Texture (View 0)

Encoding Order	POC	Picture Type	Depth(QP=32) & Colour(QP=14)		
			Bits to be Merge	Bits Merged	Bits Left
0	0	I	8720	5059	3661
1	1	P	7784	4144	3640
2	2	P	5144	4357	787
3	3	P	6264	4117	2147
4	4	P	5008	4031	977
5	5	P	5416	4073	1343
6	6	P	6112	4049	2063
7	7	P	5856	4202	1654
8	8	P	6352	4129	2223
9	9	P	6264	4067	2197
10	10	P	6816	4167	2649
11	11	P	6256	4304	1952
12	12	P	7432	4134	3298
13	13	P	6800	4089	2711
14	14	P	9512	4046	5466
TOTAL			**99736**	**62968**	

5.2.2 Sequence : Nokia_Dancer

Table 9 : Steganography Results for Nokia_Dancer sequence

QP		BITRATE(kbits/sec)	PSNR(Y) Color (dB)
14 , 32	View 0	4090.72	41.3389
	View 1	3296.5733	41.2969
	Overall	**7387.2933**	**41.3179**
18 , 36	View 0	2556.2667	33.6166
	View 1	2018.8	33.5834
	Overall	**4575.0667**	**33.6**
22 , 40	View 0	1482.96	31.2187
	View 1	1163.5467	31.2411
	Overall	**2646.5067**	**31.2299**
26 , 44	View 0	798.7733	31.2013
	View 1	612.2667	31.2199
	Overall	**1411.04**	**31.2106**

Table 10 : ”Depth as Views” Results for Nokia_Dancer sequence

QP		BITRATE(kbits/sec)	PSNR(Y) (dB)
14	View 0 C	4054.7067	47.8606
32	View 1 D	35.04	43.4652
14	View 2 C	3270.7467	47.8346
32	View 3 D	26.24	43.4534
	Overall	**7386.7334**	**45.6535**
18	View 0 C	2530.9733	44.2423
36	View 1 D	24.2267	40.8773
18	View 2 C	2002.3867	44.3525
36	View 3 D	17.68	40.5001
	Overall	**4575.2667**	**42.4931**
22	View 0 C	1467.7467	40.8741
40	View 1 D	17.08	38.3999
22	View 2 C	1154.76	40.9983
40	View 3 D	11.3733	37.7795
	Overall	**2650.96**	**39.5130**
26	View 0 C	789.44	37.8104
44	View 1 D	12.3333	35.9718
26	View 2 C	607.5067	37.9802
44	View 3 D	7.76	35.2444
	Overall	**1444.5244**	**36.7517**

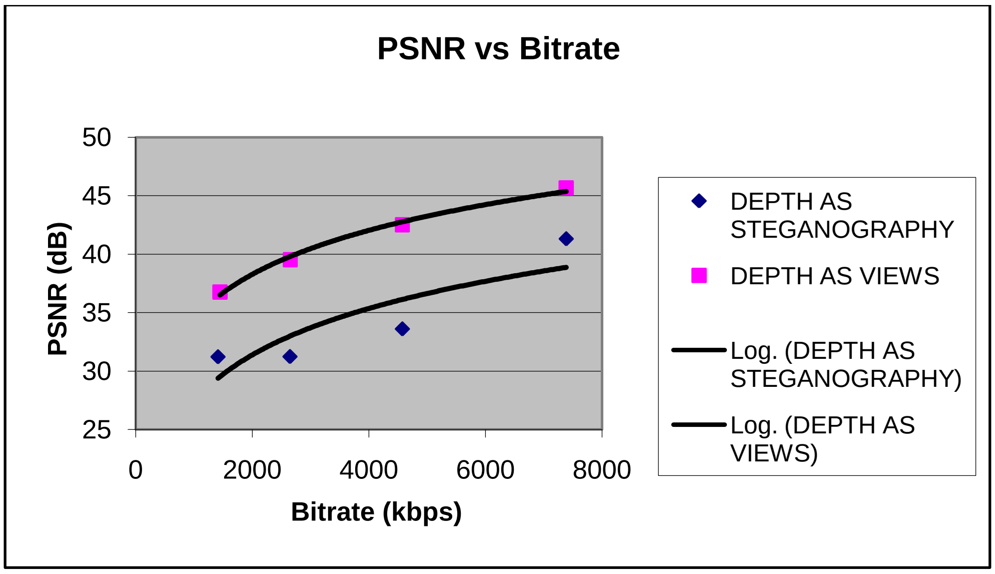


Figure 5.2 : Bitrate vs PSNR graph for Nokia_Dancer sequence

Table 11 : Merge Statistics for Nokia_Dancer sequence with same QP for Depth & Texture (View 0)

Encoding Order	POC	Picture Type	Depth(QP=32) & Color(QP=14)		
			Bits to be Merge	Bits Merged	Bits Left
0	0	I	4352	4352	0
1	1	P	3512	3512	0
2	2	P	472	472	0
3	3	P	672	672	0
4	4	P	632	632	0
5	5	P	824	824	0
6	6	P	768	768	0
7	7	P	664	664	0
8	8	P	808	808	0
9	9	P	1008	1008	0
10	10	P	960	960	0
11	11	P	872	872	0
12	12	P	1144	1144	0
13	13	P	1120	1120	0
14	14	P	4104	4067	37
TOTAL			**21912**	**21875**	

5.2.3 Sequence : MSR_Ballet

Table 12 : Steganography Results for MSR_Ballet sequence

QP		BITRATE(kbits/sec)	PSNR(Y) Colour (dB)
14 , 32	View 0	1230.8	38.9562
	View 1	1050.7067	38.7996
	Overall	**2281.5067**	**38.8779**
18 , 36	View 0	677.6667	31.8537
	View 1	561.3067	32.1951
	Overall	**1238.9734**	**32.0244**
22 , 40	View 0	414.6267	34.57
	View 1	331.2533	34.4266
	Overall	**745.88**	**34.4983**
26 , 44	View 0	253.9867	30.5057
	View 1	197.64	30.8101
	Overall	**451.6267**	**30.6579**

Table 13 : ”Depth as Views” Results for MSR_Ballet sequence

QP		BITRATE(kbits/sec)	PSNR(Y) (dB)
14	View 0 C	1178.28	48.7304
32	View 1 D	141.5867	40.1363
14	View 2 C	1002.9333	48.5378
32	View 3 D	113.4667	40.2883
	Overall	**2436.2667**	**44.4232**
18	View 0 C	651.12	46.586
36	View 1 D	87.04	37.5071
18	View 2 C	538.7067	46.4956
36	View 3 D	68.04	37.3882
	Overall	**1344.9067**	**41.994225**
22	View 0 C	398.8267	44.4005
40	View 1 D	56.4	34.9528
22	View 2 C	319.0667	44.2457
40	View 3 D	42.4267	34.6551
	Overall	**816.7201**	**39.563525**
26	View 0 C	244.9467	41.8053
44	View 1 D	36.76	31.4933
26	View 2 C	190.36	41.7165
44	View 3 D	27.9867	31.5911
	Overall	**500.0534**	**36.65155**

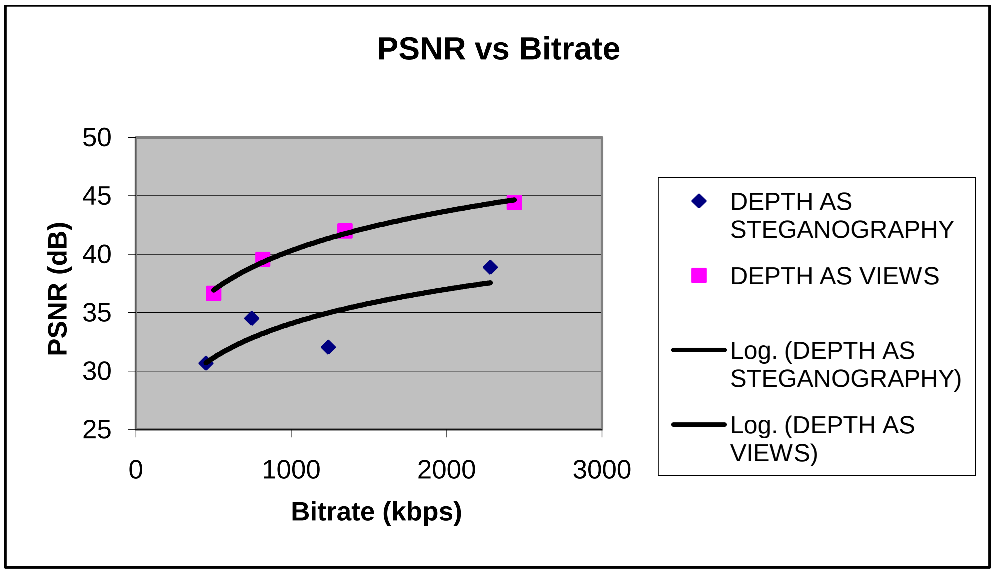


Figure 5.3 : Bitrate vs. PSNR graph for MSR_Ballet sequence

Table 14 : Merge Statistics for MSR_Ballet sequence with same QP for Depth & Texture (View 0)

Encoding Order	POC	Picture Type	Depth(QP=32) & Color(QP=14)		
			Bits to be Merge	**Bits Merged**	**Bits Left**
0	0	I	11160	3985	7175
1	1	P	9480	1425	8055
2	2	P	4928	1571	3357
3	3	P	5072	3499	1573
4	4	P	5216	1582	3634
5	5	P	5464	1749	3715
6	6	P	5056	1607	3449
7	7	P	5888	1777	4111
8	8	P	5392	1691	3701
9	9	P	4880	1758	3122
10	10	P	4488	1558	2930
11	11	P	4288	1271	3017
12	12	P	3720	1087	2633
13	13	P	4176	1315	2861
14	14	P	8160	1435	6725
TOTAL			**87368**	**27310**	

5.2.4 Sequence : Kendo

Table 15 : Steganography Results for Kendo sequence

QP		BITRATE(kbits/sec)	PSNR(Y) Colour (dB)
14 , 32	View 0	1979.2	36.2328
	View 1	1893.36	36.1159
	Overall	**3872.56**	**36.17435**
18 , 36	View 0	1173.32	29.0859
	View 1	1081.1067	29.0292
	Overall	**2254.4267**	**29.05755**
22 , 40	View 0	717.56	32.9998
	View 1	630.4267	33.0978
	Overall	**1347.9867**	**33.0488**
26 , 44	View 0	433.6267	33.4408
	View 1	360.16	33.4097
	Overall	**793.7867**	**33.42525**

Table 16 : ”Depth as Views” Results for Kendo sequence

QP		BITRATE(kbits/sec)	PSNR(Y) (dB)
14	View 0 C	1896.9067	49.1125
32	View 1 D	97.2533	42.5859
14	View 2 C	1811.6533	49.0367
32	View 3 D	165.6532	40.6532
	Overall	**3971.4665**	**45.347075**
18	View 0 C	1123.36	46.7199
36	View 1 D	60.6933	40.2689
18	View 2 C	1032.4133	46.6379
36	View 3 D	101.64	38.0208
	Overall	**2318.1066**	**42.911875**
22	View 0 C	686.2133	44.309
40	View 1 D	37.7867	37.2839
22	View 2 C	600.8533	44.2109
40	View 3 D	61.2	35.1779
	Overall	**1386.0533**	**40.245425**
26	View 0 C	415.44	41.736
44	View 1 D	22.6933	33.9667
26	View 2 C	342.88	41.5991
44	View 3 D	38.0533	32.1861
	Overall	**819.0666**	**37.371975**

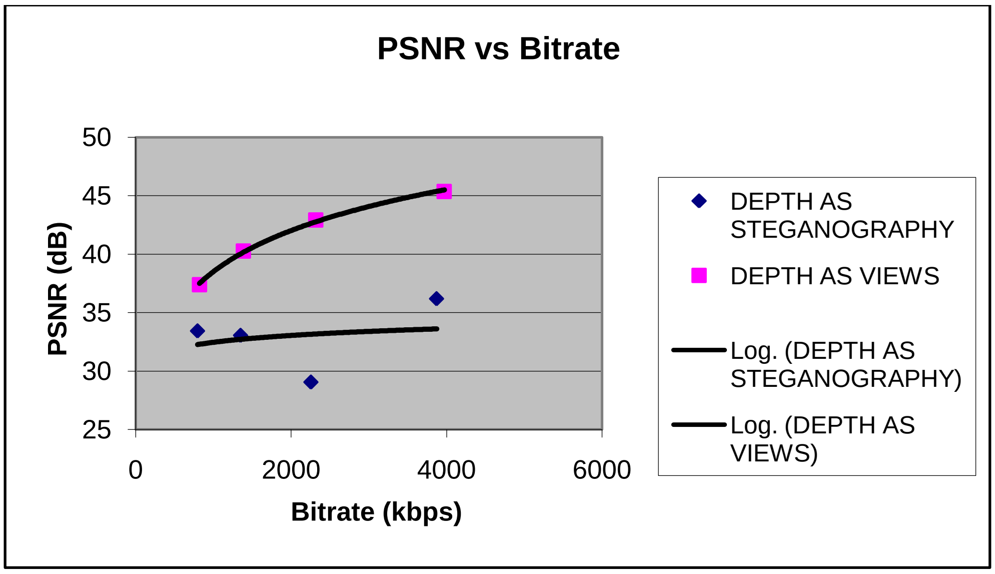


Figure 5.4 : Bitrate vs PSNR graph for Kendo sequence

Table 17 : Merge Statistics for Kendo sequence with same QP for Depth & Texture (View 0)

Encoding Order	POC	Picture Type	Depth(QP=16) & Color(QP=16)		
			Bits to be Merge	**Bits Merged**	**Bits Left**
0	0	I	25872	3741	22131
1	8	P	25712	2754	22958
2	4	B	16848	866	15982
3	2	B	13800	412	13388
4	6	B	12864	651	12213
5	1	B	11912	161	11751
6	3	B	10384	191	10193
7	5	B	10312	256	10056
8	7	B	11224	296	10928
9	12	P	17384	1424	15960
10	10	B	13232	682	12550
11	14	P	14080	1276	12804
12	9	B	11296	341	10955
13	11	B	11352	385	10967
14	13	B	11504	460	11044
TOTAL			**217776**	**13896**	

5.3 Analysis

Firstly, both texture as well as depth is encoded at the same QP, the result shown above shows very less amount of merging of encoded depth bits in the colour frame. The reason for this is discussed above i.e.

1. Skipping of macro-blocks
2. No merging due to skipped blocks because of invalid coded block pattern,
3. No merging should take place when number of significant coefficient is equal to zero, which means no non zero coefficient present in a block
4. No merging should take place when after merging the last level coefficient value of a block is changing from "1" to "0". Merging is avoided here to keep uiNumSigCoeff unchanged and to provide seamless decoding.

In section 5.2, different sequences are encoded with the provision that Depth frames are encoded at higher QP than that of colour frames. Results in section 5.2 shows that out of four different sequences, maximum merging (almost all bits) takes place in Nokia_Dancer sequence.

For each sequence, Bitrate vs. PSNR graph is shown, and as shown in graphs we are getting less amount of gain in bitrate and significant decrease in PSNR. In the range, approximately 6dB-12dB fall in PSNR is noticed, this is a remarkable decrease in the PSNR i.e. objective quality of the video and could be fixed.

Chapter 6

Thesis Conclusion

6.1 Conclusion

The main contribution of this work is that we had developed and implemented an algorithm which is used to compress the data in which instead of separate bit-streams each for depth and colour, only one bit-stream for colour (containing depth data) is developed. And this implementation works not only on same QP for both texture and depth but also on the different QP for texture and depth. We also got the different reasons for less merging of encoded depth data bits into the colour frame, and this varies sequence to sequence.

6.2 Future Directions

As we saw that there is a remarkable decrease in PSNR i.e. quality of video after decoding the bitstream which we got after using steganography approach. So, some modifications in the current implementation or new approach need to be followed.

One method can be “bit domain labelling” [32] in which the merging of encoded depth bits into colour frame is done by selecting suitable VLC’s instead of merging in all of them. It is shown in [32] that to ensure the change in the VLC yields perceptually invisible degradation after decoding, we select only those VLC's for which another VLC exists with the same run length, a level difference of one and the same codeword length.

And another method to cure the problem of bits left during merging can be Down-sampling the depth map and keeping the texture frame at resolution greater than that of depth frame. By down-sampling the number of encoded depth data bits become less and can easily be accommodated in colour frames during encoding.

Bibliography

[1] C. Wheatstone, "Contributions to the physiology of vision - part the first. on some remarkable, and hitherto unobserved phenomena of binocular vision," *Philosophical Transactions*, vol. 128, pp. 371–394, 1838.

[2] I. Sexton and P. Surman, "Stereoscopic and auto-stereoscopic display systems," *IEEE Signal processing magazine*, vol. 16, pp. 85–99, May 1999.

[3] A. Redert, R.-P. Berretty, C. Varekamp, O.Willemsen, J. Swillens, and H. Driessen, "Philips 3D solutions: From content creation to visualization," in *Proceedings of the Third International Symposium on 3D Data Processing, Visualization, and Transmission*, (Chapel Hill, USA), pp. 429–431, 2006.

[4] J. Ilgner, J. J.-H. Park, D. Labb´e, and M. Westhofen, "Using a high definition stereoscopic video system to teach microscopic surgery," in *Proceedings of the SPIE, Stereoscopic Displays and Virtual Reality Systems XIV*, vol. 6490, (San Jose, USA), p. 649008, February 2007.

[5] C. L. Zitnick, S. B. Kang, M. Uyttendaele, S. Winder, and R. Szeliski, "High-quality video view interpolation using a layered representation," *ACM Transactions on Graphics*, vol. 23, no. 3, pp. 600–608, 2004.

[6] I. E. G. Richardson. H.264 and MPEG-4 Video Compression: Video Coding for Next Generation Multimedia. Wiley, UK, 2003.

[7] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra. Overview of the H.264/AVC video coding standard. *IEEE Trans. Circuits Syst. Video Technol*., 13(7):560-576, July 2003.

[8] ISO/IEC 14 496-10 and ITU-T Rec. H.264, Advance Video Coding, 2003.

[9] ISO/IEC 14496-2 (MPEG-4 Part 2), Coding of audio-visual objects Part 2: Visual, January 1999.

[10] ITU-T Rec. H.262 and ISO/IEC 13818-2 (MPEG-2), Generic coding of moving pictures and associated audio information Part 2: Video, Novembor 1994.

[11] V. Bhaskaran and K. Konstantinides. Image and Video Compression Standards, Algorithms, and Architectures. 2nd edition, Kluwer, USA, 1999.

[12] ISO/IEC 15444-3 Motion-JPEG2000 (JPEG2000 Part 3), 2002.

[13] ISO/IEC 15444-1 JPEG2000 and ITU-T Rec. T.800, Image Coding System: Core Coding System (JPEG2000 Part 1), 2000.

[14] A. Joch, F. Kossentini, H. Schwarz, T. Wiegand, and G. J. Sullivan. Performance comparison of video coding standards using lagragian coder control. In *Proceedings of IEEE Int. Conf. Image Processing*, pages 501-504, September 2002.

[15] M. Kim, H. Lee, and S. Sull. Spatial error concealment for H.264 using sequential directional interpolation. *IEEE Trans. on Consumer Electronics*, 54(4):1811-1818, November 2008.

[16] MPEG-4 part 10 AVC (H.264) video encoding, June 2005. Scientific Atlanta, Inc.

[17] G. J. Sullivan, P. Topiwala, and A. Luthra. The H.264/AVC advanced video coding standard: overview and introduction to the Fidelity range extension. In *Proceedings of SPIE on Applications of Digital Image Processing*, volume 5558, November 2004.

[18] J. S. Park and H. J. Song. Selective intra prediction mode decision for H.264/AVC encoders. In *Proceedings of World Academy of Science, Engineering, and Technology*, volume 13, pages 1307-6884, May 2006.

[19] F. Pan, X. Lin, S. Rahardja, K. P. Lim, Z. G. Li, and D. Wu an S. Wu. Fast mode decision algorithm for intra prediction in H.264/AVC video coding. *IEEE Trans. Circuits Syst. Video Technol*., 15(7):813-822, July 2005.

[20] A. Puria, X. Chenb, and A. Luthra. Video coding using the H.264/MPEG- 4 AVC compression standard. *Signal Processing: Image Communication*, 9(9):793-849, October 2004.

[21] P. Yin, H. Y. Cheong, A. Tourapis, and J. Boyce. Fast mode decision and motion estimation for JVT/H264. In *Proceedings of IEEE Int. Conf. Image Processing*, 2003.

[22] C. W. Ku, C. C. Cheng, G. S. Yu, M. C. Tsai, and T. S. Chang. A High definition H.264/AVC intra-frame codec IP for digital video and still camera applications. *IEEE Trans. Circuits Syst. Video Technol*., 16(8):917-928, August 2006.

[23] H. S. Malvar, A. Hallapuro, M. Karczewicz, and L. Kerosfsky. Low-complexity transform and quantization in H.264/AVC. *IEEE Trans. Circuits Syst. Video Technol*., 13(7):598-603, July 2003.

[24] D. Marpe, H. Schwartz, and T. Wiegand. Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard. *IEEE Trans. Circuits Syst. Video Technol*., 13(7):620-636, July 2003.

[25] R. R. Osorio and J. D. Bruguera. High-throughput architecture for H.264/AVC CABAC compression system. *IEEE Trans. Circuits Syst. Video Technol*., 16(11):1376-1384, November 2006.

[26] G. Raja and M. J. Mirza. In-loop deblocking filter for H.264/AVC video. In *Proceedings of the ISCCSP*, March 2006.

[27] P. List, A. Joch, J. Lainema, G. Bjontegaard, and M. Karczewicz. Adaptive deblocking filter. *IEEE Trans. Circuirs Syst. Video Technol*., 13(7):614-619, July 2003.

[28]. L. Stelmach and W. J. Tam, ” Stereoscopic Image Coding: Effect of Disparate Image-Quality in Left- and Right-Eye Views,” *Signal Processing: Image Communication* , Vol. 14, pp. 111-117, 1998

[29]. L. Stelmach, W.J. Tam; D. Meegan, and A. Vincent, “Stereo image quality: effects of mixed spatio-temporal resolution,” *IEEE Trans. Circuits and Systems for Video Technology,* Vol. 10, No. 2, pp. 188-193, March 2000.

[30]. P. Merkle, A. Smolic, K. Mueller, and T. Wiegand, “Efficient Prediction Structures for Multiview Video Coding,”*IEEE Trans. Circuits and Systems for Video Technology*, Vol. 17, No. 11, Nov. 2007

[31] A. Vetro, P. Pandit, H. Kimata, A. Smolic, and Y.-K.Wang, “Joint draft 8.0 on multiview video coding.” Joint Video Team (JVT) of ISO/IEC MPEG ITU-T VCEG ISO/IEC JTC1/SC29/WG11 and ITU-T SG16 Q.6, July 2008.

[32] C. L. Gerrit, L. L. Reginald, Jan Biemond, “Real Time Labeling of MPEG-2 Compressed Video” *Journal of Visual Communication and image representation,* Vol. 9, No. 4, pp. 256-270, Dec. 1998.