DEVELOPMENT OF COMPRESSION METHODOLOGY FOR 3D VIDEO

By

Parth D. Desai

10MCEC03

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

AHMEDABAD-382481

May 2012

DEVELOPMENT OF COMPRESSION METHODOLOGY FOR 3D VIDEO

Major Project

Submitted in partial fulfillment of the requirements

For the degree of

Master of Technology in Computer Science and Engineering

By

Parth D. Desai

10MCEC03

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

AHMEDABAD-382481

May 2012

Declaration

I, **Parth D. Desai**, **10MCEC03**, give undertaking that the Major Project entitled **"Development of Compression Methodology for 3D Video"** submitted by me, towards the partial fulfillment of the requirements for the degree of Master of Technology in Institute of Technology of Nirma University, Ahmedabad, is the original work carried out by me and I give assurance that no attempt of plagiarism has been made. I understand that in the event of any similarity found subsequently with any published work or any dissertation work elsewhere; it will result in severe disciplinary action.

Parth D. Desai

Certificate

This is to certify that the Major Project entitled "DEVELOPMENT OF COMPRESSION METHODOLOGY FOR 3D VIDEO" submitted by Parth D. Desai (10MCEC03), towards the partial fulfillment of the requirements for the degree of Master of Technology in Computer Science and Engineering, Nirma University, Ahmedabad is the record of work carried out by him under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this major project, to the best of my knowledge, haven't been submitted to any other university or institution for award of any degree or diploma.

Mr. Srijib Narayan Maiti
Major Guide and Employee,
Advance System Technology,
ST Microelectronics,
Greater Noida

Prof. Samir Patel
Internal Guide and Associate Professor
Dept. of Computer Engineering,
Institute of Technology,
Nirma University, Ahmedabad

Dr. S.N. Pradhan
Professor and PG-Coordinator,
Dept. of Computer Engineering,
Institute of Technology,
Nirma University, Ahmedabad

Prof D.J. Patel
Professor and Head of the Dept.
Dept. of Computer Engineering,
Institute of Technology,
Nirma University, Ahmedabad

Dr. Ketan Kotecha
Director,
Institute of Technology,
Nirma University, Ahmedabad

Abstract

In the current thesis, a method to encode/decode multiview textures and associated depth maps together in a single bitstream is proposed. A methodology which uses Steganography approach in video compression is designed and implemented to compress the data in which instead of separate bit-streams each for depth and texture, only one bit-stream for texture (containing depth data) is developed. First, depth map is downsampled and then the encoded bits of depth maps have been merged into that of associated textures. Encoded bits of depth maps have been merged into that of associated textures in an innovative way with very minimal loss in PSNR of textures, 0.1 - 0.2 dB for high delay and 0.2 - 0.5 dB for low delay applications with overall bitrate gain of about 17% for high delay and 13% for low delay applications and for natural scenes. The encoding technique ensures backward compatibility (in texture only decode, e.g. MVC stereo high profile) and correct extraction of downsampled encoded depth maps followed by decoding of downsampled depth maps. Benchmarking results in terms of compression efficiency and quality on HD sequences have been reported to explore further the viability of using this technique for Free-view point Video (FVV) or 3DV use cases. Experiments are also carried out to evaluate subjective quality of the synthesized views.

Acknowledgements

With immense pleasure I express my sincere gratitude, regards and thanks to my thesis supervisor and major guide **Mr. Srijib Narayan Maiti** for his constant guidance, motivation and encouragement at all the stages of my research work. He has devoted significant amount of his valuable time to plan and discuss the thesis work. Without his experience and insights, it would have been very difficult to do quality work. I would like to thank my project manager **Mr. Chander Kaher** and all my teammates at ST Microelectronics especially **Emiliano Mario Piccinelli, Davide Aliprandi** and all others for their support and help.

I wish to place on record my gratitude to **ST Microelectronics**, Greater Noida for providing me an opportunity to work with them on this project of such importance. My stay in the organization has been a great learning experience and a curtain raiser to an interesting and rewarding career. This exposure has enriched me with technical knowledge and has also introduced me to the attributes of a successful professional.

I would also like to extend my gratitude to my Internal Guide **Prof. Samir Patel** for his guidance and fruitful suggestions. My deep sense of gratitude to **Dr. S.N. Pradhan**, Professor and PG-Coordinator of Department of Computer Engineering, Institute of Technology, Nirma University, Ahmedabad for an exceptional support and continual encouragement throughout the tenure of the Major project. I would like to thank **Dr. Ketan Kotecha**, Hon'ble Director, Institute of Technology, Nirma University, Ahmedabad for his unmentionable support, providing basic infrastructure and healthy research environment.

I would also thank my Institution, all my faculty members in Department of Computer Science and my colleagues without whom this project would have been a distant reality. Last, but not the least, no words are enough to acknowledge constant support and sacrifices of my family members because of whom I am able to complete the degree program successfully.

- Parth D. Desai
10MCEC03

Contents

List of Tables

List of Figures

Abbreviation Notation and Nomenclature

JPEG	Joint Picture Experts Group
MPEG	Moving Picture Experts Group
VCEG	Video Coding Experts Group
FTV	Free Viewpoint Television
DCT	Discrete Cosine Transform
VCL	Video Coding Layer
NAL	Network Adaptation Layer
AVC	Advance View Coding
MVC	Multi View Coding
PSNR	Peak Signal to Noise Ratio
SSIM	Structural Similarity Index Metric
MB	Macro Block
QP	Quantization Parameter
CAVLC	Context Adaptive Variable Length Coding
CABAC	Context Adaptive Binary Arithmetic Coding
ME	Motion Estimation
MV	Motion Vector
MVVD	Multi View Video and Depth
SPS	Sequence Parameter Set
JMVC	Joint Multi View Coding
JVT	Joint Video Team
IDE	Integrated Development Environment
HEVC	High Efficient Video Coding

Chapter 1

Introduction

1.1 Current trends in 3D Video Compression and Transmission

The display industry is witnessing a major change from 2-D to realistic 3-D as evidenced by the success of recent 3-D movies and consumer 3DTV product offerings. It can be envisioned that within the next few years a major share of current displays will be replaced by 3-D displays or at least by 3-D-capable displays and stereoscopic and auto-stereoscopic technologies are likely to play a major role. 3-D display users are unlikely to accept a compromise on image quality and resolution while switching from 2-D to 3-D displays. Currently, stereoscopic direct view technology is used for 3D video display. These systems use stereo video coding for pictures delivered by two input cameras. Such stereo systems only reproduce these two camera views at the receiver and stereoscopic displays for multiple viewers require wearing special 3-D glasses. But still this type of transmission is under commencing phase and is expected to dominate the domain of 3D video transmission in next few years. In future, auto-stereoscopic 3D display technology shall be used. These emerging auto-stereoscopic multiview displays emit a large numbers of views to enable 3-D viewing for multiple users without requiring 3-D glasses. For each number of views there will be the associated depth maps to perceive the 3D effect. For representing a large number of views, a multiview extension of stereo video coding is used, typically requiring a bit rate that is proportional to the number of views. Presently, the transmission of such types of

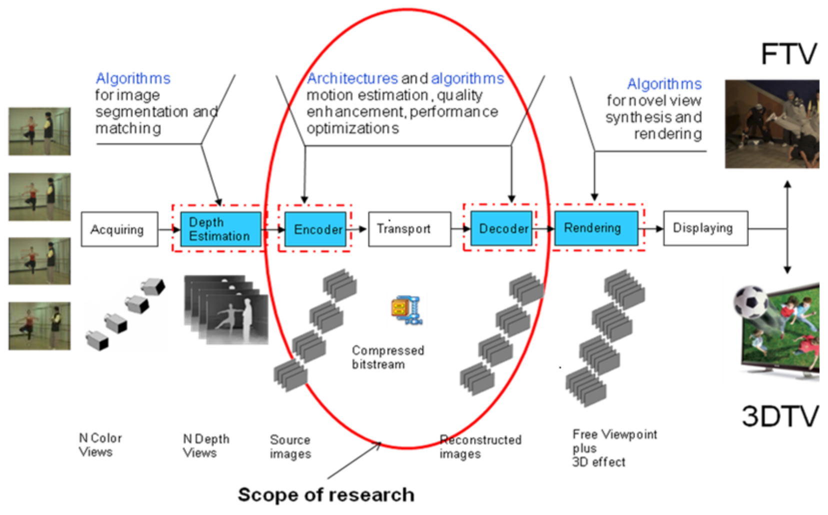
3D video is still not available due to high requirement of bandwidth.

However, a major problem when dealing with multi-view video is the intrinsically large amount of data to be compressed, decompressed and rendered. So we concentrate on the compression problem of multi-view video and depth coding. We propose a method for 3D video compression, in which the depth information is hidden in the information of texture using steganography approach. This thesis report is based on the study of video plus depth compression and its development by modifying the available MPEG-Part 10 H.264 video coding standard and its optimization for 3D services.

1.2 Motivation

There have been many efforts reported in the past, where compression of the 2D video is implemented using various standards. But the field of 3D Video is still not explored to that extent. After the success of stereoscopic 3D, there has been an increased interest for more immersive experiences for television watching. This can be achieved by capabilities of display devices to render multiple views on the 3D display [5]. As an ultimate experience beyond just 3DTV, a new immersive visual media has been emerged as Free-Viewpoint Television (FTV) which enables user to freely choose the view point of a 3D scene [6]. Due to the rapid growth of 3D TV and 3D consumer electronic products, the issue of compressed stereo images for easier transmission and storage is attracting much attention. In addition to stereo displays, also multiview displays are becoming increasingly available. Due to the huge amount of data in multiview video coding, particularly when the number of views decoded is large, transmission of multiview video applications relies heavily on the compression of the video captured by cameras. Therefore, efficient compression of multiview video contents is the primary challenge for realizing multiview video services. Presently, there is no efficient technology that transmits multiple views using multiview coding (MVC) profile by maintaining the high quality of the video. So this motivated me to prepare my thesis topic on 3D Video Compression for MVC.

1.3 Scope of Research



Algorithms for image segmentation and matching
Architectures and algorithms motion estimation, quality enhancement, performance optimizations
Algorithms for novel view synthesis and rendering
FTV
Acquiring
Depth Estimation
Encoder
Transport
Decoder
Rendering
Displaying
Compressed bitstream
3DTV
N Color Views
N Depth Views
Source images
Reconstructed images
Free Viewpoint plus 3D effect
Scope of research

Figure 1.1: An example of FTV scenario[2]

The scope of the work, as highlighted in Figure 1, is to develop an efficient mechanism to encode and decode multiple views and the associated depth maps which is one of the main issues in the whole chain. The exploration is done at the encoding phase and the decoding phase for multi view video. Thus, the main goal of the thesis project is to transmit the multiple views along with associated depth maps with minimum requirement of bandwidth.

1.4 Thesis Organization

The rest of the thesis is organized as follows.

Chapter 2, *Literature Review*, describes the literature survey and the design goal of the thesis. Exploration is done to get the understanding of the problem and to get a brief idea of the existing compression techniques.

Chapter 3, *The Proposed Methodology*, A new methodology is proposed using Steganography approach is proposed to compress the auto stereoscopic 3D data in which instead of separate bit-streams each for depth and texture view, only one bit-stream for texture (containing depth data) is generated. Complete methodology and concept of merging is described.

Chapter 4, *Implementation of the Proposed Techniques*, The complete implementation of the proposed methodology using open source software jmvc8.0 is presented.

Chapter 5, *Simulation Results and Analysis*, The simulation results that are carried out using the proposed methodology and its complete analysis is been discussed.

Finally, in **Chapter 6**, *Conclusion*, The conclusion of the thesis work is presented.

Chapter 2

Literature Review

2.1 Existing 3D Video Coding Formats

The most obvious and straightforward means to represent stereo or multi-view video is simulcast, where each view is encoded independent of the other. This solution has low complexity since dependencies between views are not exploited, thereby keeping computation and processing delay to a minimum. 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 or coded with a reduced spatial resolution [13], yielding an imperceptible impact on the stereo quality.

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 [12]. The main drawback is that the format is only capable of rendering a limited depth range and has problems with occlusions.

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. 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.

Developing an efficient mechanism to encode and decode multiple views and the associated depth maps is one of the main issues in the whole chain of video coding. MPEG has been progressing towards this in two phases. In the first phase it concentrated on development of coding mechanisms and transport of multiview video over the broadcast network. This has been standardized as H.264/MVC [15]. MVC provides a compact representation for multiple views of a video scene, such as multiple synchronized video cameras. An overview of the algorithmic design used for extending H.264/MPEG-4 AVC towards MVC is discussed in [15]. The problem of MVC in respect of FTV is that the bandwidth requirements are linearly proportional to the number of views [16]. So there is a need to decouple the number of views transmitted and the actual number of views rendered at the customer premises. A call for proposals (CfP) for 3D Video (3DV) [17] has been issued in March 2011 to address these in 2nd phase. Many researchers have carried out innovative explorations for increasing the compression efficiency of the final bitstream containing multiple views and/or associated depth maps separately. NTT Corporation, MERL has shown good results utilizing view synthesis prediction (VSP) alongwith normally used disparity compensated (DCP) [17] alongwith various improvement methods like decoder side motion vector derivation, adaptive filtering to improve on VSP [19], [20]. HHI has been carrying out researches towards various aspects of FTV [21] [22] [23]. Reference [22], [23] explores a method to encode either multiple video or multiple depth maps. Disney Research Zurich describes a mechanism of generating multiple views without even using depth maps [24]. None of these are focusing on encoding of multiple textures and depth maps together in single bitstream.

NTT Corporation [25] has shown one possibility to realize FTV in real time using both CPU and GPU compressing of multiple textures and multiple depths using H.264/MVC. The advantage of this approach is limited to the efficiency of H.264/MVC. Our approach is actually using this concept and applied an innovative way to merge the encoded depth maps in the bitstream of textures reducing the overall bandwidth requirements without compromising too much on the quality. Reference [26] by NTT Corporations is also one of the methods to encode textures and depth map but this scheme is assumed to have the view synthesis running at the encoder end. HHI and Technical University of Berlin showed [27] two methods to encode multiple textures and depth maps together. One method is similar to that of [25] and the other method utilizes the 'auxiliary' picture syntax of the H.264 standard to encode depth map. Here in the scenario of multiple depth maps, we will not be able to utilize inter-view prediction for depth maps and the resolution of depth maps is always same as that of the resolution of textures.

2.2 Steganography

Steganography is a technique used to transmit the secret message to the intended receiver in such a way that apart from the sender and receiver no one other can suspect the existence of the message. In this technique, message is embedded with the source data in such way that some other person can't perceive the presence of message. Due to attachment of the secret message into the source data there are chances of alternation in the original source data. Here the original source in which message is attached could be images, videos or audio files. In the field of cryptography the technique of steganography is very popular. Here the secret message is embedded into the images. On the receiver side the receiver decode those images and can fetch the embedded message.

In most of the cases, steganography is the field in which people use images for hiding the information but this can be extended up to video or audio files. Steganography in image is implemented by merging the least significant bit (LSB) of every byte of the source image with the bits of the secret message. However in video, a minor change is made to implement the concept of steganography. Application of steganography in video has been broadly into two domains. One is in the uncompressed and the other in compressed domain. However the later is more significant when compression efficiency is of prime importance and the same is explored further in this thesis. As discussed in Stanescu et al. [28], they proposed a steganography method in which data is embedded in Intra frames of the video stream using spatial characteristics. In their algorithm, for every I-frame the DCT coefficients were calculated and the coefficients which are above the threshold value are found. The LSB of these coefficients is replaced by a bit of the secret message. Hu et al [29] have presented an algorithm that embedded one bit in each qualified intra block in H.264 bit stream. Many researchers have carried out innovative explorations for increasing the use of steganography for various applications in the areas of cryptography, compression and many more. Some researchers proposed a technique in which they exploited temporal redundancy of the frames. Xu et al.[30] proposed a technique which used temporal redundancy. Instead of merging the secret message into the DCT coefficients, they used motion vectors. The motion vectors were calculated and the motion vectors with large magnitude are selected for data hiding purpose.

When the concept of steganography is extended to video compression, two parameters become significant, number of bits merged in the source data and degradation tolerance in a source data. As discussed in Jafar et al [31], above technique has a low embedding capacity. Therefore they modified the technique and proposed an algorithm which merged the secret message bit in video based on the threshold value. In their technique, the algorithm used 8x8 blocks of image and calculated DCT coefficients. DC coefficients represents average energy of that block so it can't be used for merging secrete message while AC coefficient represent intensity changes. Therefore their algorithm used AC coefficients for hiding the information. Algorithm calculated square sum of all AC coefficients for that block and if sum exceeded the threshold value then only merging was performed. So this increased the number of bits merged as well as the quality of the video was maintained. However, in the above technique, the application of the algorithm is different. Jafar et al [31] talks about the covert channel communication application where the sole intention is sending a secret message without degrading video quality. But we will be using the steganography technique for merging the depth map into multiple views. So from our point of view the quality of the video as well as the number of depth bits merged into texture are also important. In our technique, we are not hiding any secret message to a video or audio into video but we hide encoded video into an encoded video. Therefore both the source video and the merged video are crucial for us and their PSNRs are our prime concern with appropriate compression.

2.3 H.264 Standard

Video compression (or video coding) is an essential technology for applications such as digital television, DVD-video, mobile TV, videoconferencing and internet video streaming. Standardising video compression makes it possible for products from different manufacturers (e.g. encoders, decoders and storage media) to inter-operate. The Moving Picture Experts Group and the Video Coding Experts Group (MPEG and VCEG) have developed a new standard that promises to outperform the earlier MPEG-4 and H.263 standards, providing better compression of video images. This new standard is MPEG-4 Part 10 also called H.264. H.264 is an industry standard for video compression, the process of converting digital video into a format that takes up less capacity when it is stored or transmitted. H.264 is an industry standard for video compression, the process of converting digital video into a format that takes up less capacity when it is stored or transmitted. An encoder converts video into a compressed format and a decoder converts compressed video back into an uncompressed format.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 specific transport layers or storage media)[7]. An H.264 video encoder carries out prediction, transform and encoding processes (Figure 2.1) to produce a compressed H.264 bitstream. An H.264 video decoder carries out the complementary processes of decoding, inverse transform and reconstruction to produce a decoded video sequence. The picture is divided into macroblocks that consists 16x16 samples of Luma and 8x8 samples of two Chroma components.

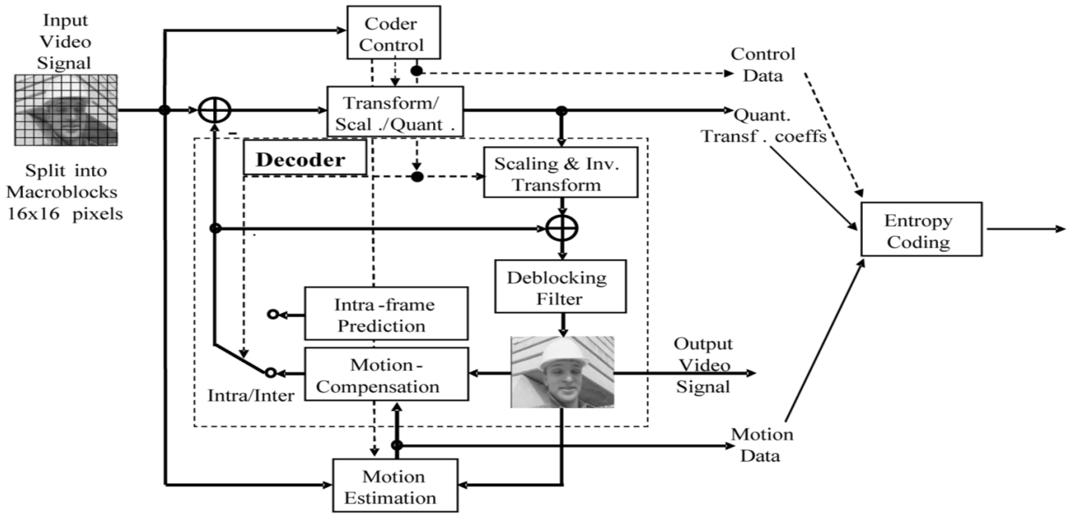


Figure 2.1: H.264/AVC Block Diagram[8]

2.3.1 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. Y is the luminance component, which represents the brightness information of the image. Cr and Cb are the chrominance components, and represent the color information of the image. There are two types of prediction: Inter Prediction and Intra Prediction.

2.3.2 Transform and Quantisation

After the intra or inter prediction, the best predicted MBs are selected. This best predicted macroblocks are subtracted from the original macroblock to form a residual. This is called Motion Compensation [10]. This residual is then transformed and quantized. A block of residual samples is transformed using a 4x4 integer transform, an approximate form of the Discrete Cosine Transform (DCT) [10]. The transform outputs a set of coefficients, each of which is a weighting value for a standard basis pattern. When combined, the weighted basis patterns re-create the block of residual samples. The output of the transform, a block of transform coefficients, is quantized, i.e. each coefficient is divided by an integer value [10]. 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. Quantization reduces the precision of the transform coefficients according to a quantization parameter (QP).

2.3.3 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) and context adaptive binary arithmetic coding (CABAC) are used [9]. The CABAC mode offers approximately 10%-15% enhanced compression efficiency compared to the CAVLC mode. For the current project, CABAC shall be used as entropy coding technique.

2.3.4 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 [11]. 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 [11]. 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.

2.3.5 Design Goal

The transmission and compression of multiview 3D video possess significant challenges. First, it is bandwidth-intensive, as it requires the transmission of multiple large-volume 3D video streams. Second, existing schemes for 2D texture video compression such as MPEG, JPEG, and H.263 cannot be applied directly because the 3D video data contains depth as well as texture information. My goal is to explore from a different angle of the auto stereoscopic 3D compression space with factors including complexity, compression ratio, quality, and real-time performance. To investigate these trade-offs, I evaluated two 3D compression techniques for a multi view video system. The first technique addresses the problem of 3D multi-view video acquisition by multiview geometry and coding. Multi-view video acquisition refers to the task of estimating and recording a 3D geometric description of the scene based on the extrinsic camera parameters and the camera positions. Here instead of sending the depth map, the depth map is itself generated at the decoder. Only one proposal has been made describing this technique. The second technique is based on the compression problem of multi-view texture and depth video. Here, the depth maps need to be encoded and send to the decoder which will decode the depth maps. 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 numbers of views.

Chapter 3

The Proposed Methodology

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 texture, only one bit-stream for texture (containing depth data) is developed.

3.1 The Problem

Currently, stereoscopic direct view technology is used for 3D video display. This technology is based on stereo formats, where the two views are coded using the stereo high profile of H.264/AVC. The extension of the stereo high profile to many views is called MVC. For representing a large number of views, a multiview extension of stereo video coding is used, typically requiring a bit rate that is proportional to the number of views. Presently, the transmission of such types of 3D video is still not available due to high requirement of bandwidth. So in order to compress the auto stereoscopic 3D video, the depth and texture information is to be encoded separately and the bitstream for depth and texture is generated. In this way a good amount of compression is achieved. But the compression achieved is still not satisfactory. The compressed video output that is achieved using H.264/MVC can still be further compressed. Our main goal is to generate a single bitstream which consists of texture and depth information in the single bitstream. This approach can be extended for multiple views using H.264/MVC.

3.2 Our Proposal and Method

Our method of generating single bitstream for multiple textures and depth maps employs merging the bits of encoded depth maps into that of corresponding textures using the concept of steganography. This process (merging) is applied at residuals of each block, i.e. either 4x4 or 8x8 levels, depending on the transform size being used at macroblocks. So, while encoding each frame of a particular view, associated depth map is encoded first but not included in the final bitstream. Then texture is encoded and before the final CAVLC/CABAC stage of texture encoding the encoded depth bits is merged one by one at least significant bit (LSB) position of the last level of each residual block. So, essentially the depth maps are inserted into the textures as noise. At the decoder side the inserted depth maps are extracted and then decoded followed by decoding of texture. As a result, quality of texture is degraded but no degradation of depth map can be observed. In following section we discuss our method in more details taking H.264/MVC as the basic infrastructure of coding. Figure 3.1 shows the block diagram of the encoder where merging is implemented for view 0 (V0) and associated depth map i.e. depth 0 (D0).

3.2.1 Merging in final bitstream

Before merging the depth bits into texture, first the depth is to be encoded. The idea is that, we need to have depth encoded bit stream for each frame so that we can merge this bitstream into the texture encoded bits for the respective frame. Figure 3.1 shows the block diagram of the encoder where merging is implemented for view 0 (V0) and associated depth map i.e. depth 0 (D0). First, each frame of depth is predicted, transformed, quantized and encoded to form the bitstream of depth. Now, for each texture frame, 4x4 blocks are transformed and quantized and the coefficients are generated. These coefficients are then reordered in zigzag scan fashion and the last significant coefficient is found. Then, one bit of encoded depth is merged with the LSB of the last significant coefficient. Rest of the process is as usual as that of H.264/AVC and is similarly implemented for other enhanced views and associated depth maps too. In brief, the "MERGING" block is only addition in the encoding process which is explained in next section.

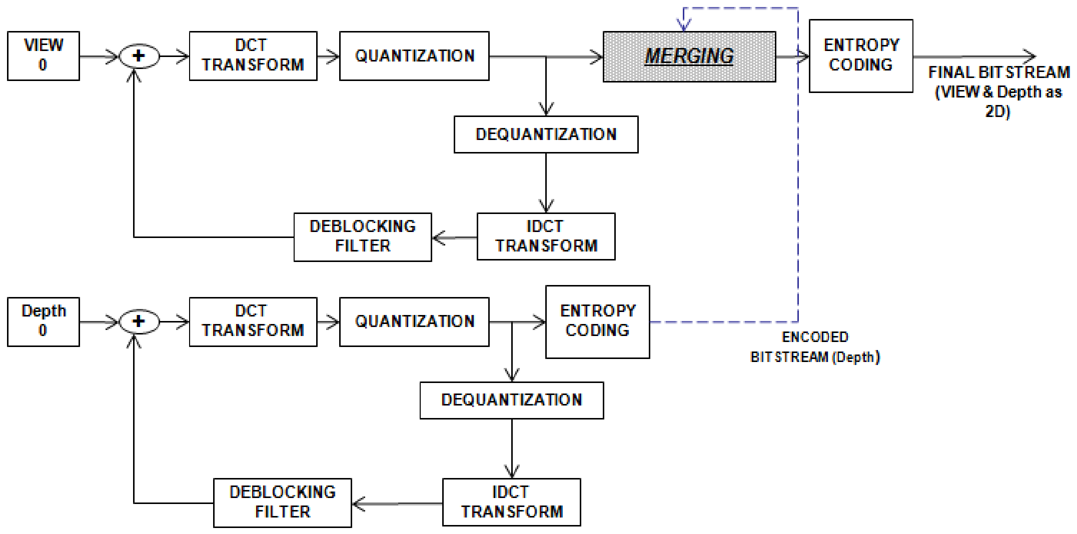


Figure 3.1: Merging in final bitstream in H.264/MVC Encoder for mono view

3.2.2 Merging Concept

The merging of the encoded depth bits into the coefficients of texture is shown in Figure 3.2. The depth bitstream is embedded in texture bitstream by replacing the LSB of the quantized last significant coefficient with a bit of depth bitstream. 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. To add the depth bitstream to a texture 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 Li (I = 0,1,2,, l), this LSB is replaced by the depth bit Li . And if the LSB of its level equals the depth bit Li the LSB is not changed. The procedure is repeated until all Depth bits are merged/embedded. The procedure is repeated until all depth bits are merged/embedded. The above procedure is repeated for every block until all bits of the encoded depth map, henceforth called the ’message’, are inserted into the texture bitstream. In this way a depth bit is hidden in texture bitstream. The insertion of the depth bits in texture is not visible because human eye cannot detect negligible change of colour. The merging is not implemented for the DC coefficients in intra blocks because they are predicted from other DC coefficients and are very significant in terms of containing valuable information.

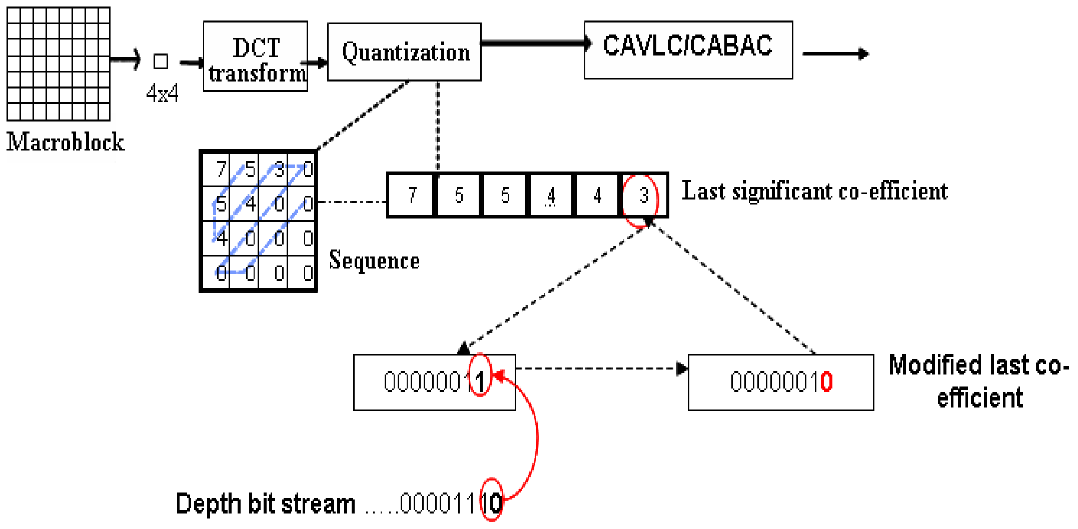


Figure 3.2: Merging of a depth bit at coefficients of texture

From results shown in Figure 5.1, we can derive that there is a significant drop in PSNR (8 dB to 10 dB) compared to the "Anchor". Thus the quality of picture is degraded in a visible manner which is unacceptable. Reason behind this degradation is that motion estimation loop at the encoder doesn't compensate for the degradation in the final bitstream since the encoder doesn't have any feedback for the modification at the final bitstream. So at the decoder side the error keeps on propagating which in turn results to a high loss in PSNR. To avoid this problem we implemented merging inside the motion estimation (ME) loop as discussed next.

3.3 Merging in Motion Estimation Loop

Applying merging outside the motion estimation loop lead to the high degradation in the quality of texture. To avoid this problem, a small modification in the approach was made and we implemented merging inside the motion estimation (ME) loop. While implementing the merging in motion estimation loop, the cost and the distortion rate shall be calculated based on the merged coefficients and the best mode shall be selected based on it. Figure 3.3 shows the merging implemented in motion estimation loop.

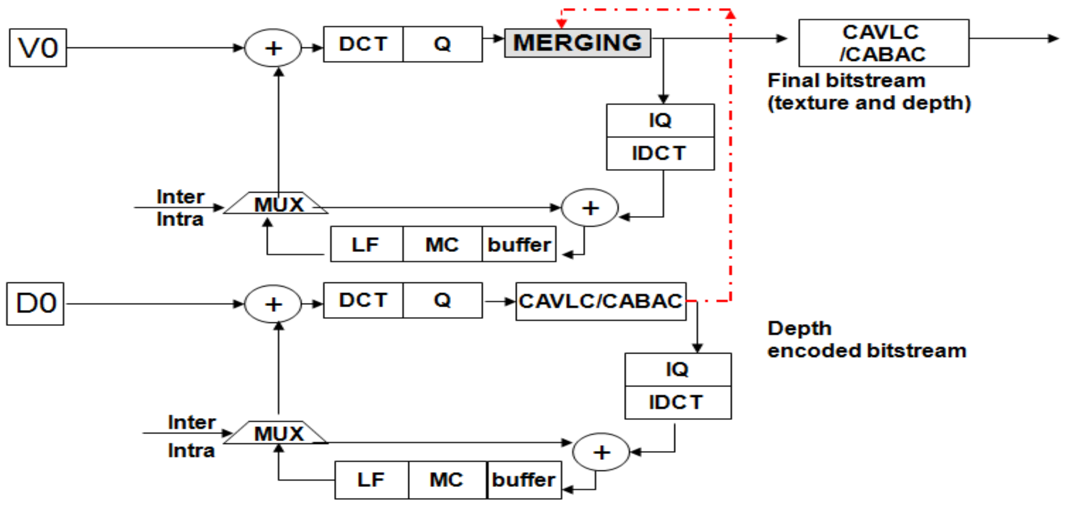


Figure 3.3: Merging approach in Motion Estimation loop in H.264/MVC Encoder for mono view

While moving the merging block inside the motion estimation loop, it was mandatory that the merging block has to be implemented in all the prediction modes including for I, P and B frames. So, merging block was implemented in all the following prediction modes.

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. There are two types of prediction: Intra Prediction and Inter Prediction.

3.3.1 Intra Frame Prediction

In Intra prediction mode, a MB is predicted from the neighboring previously decoded and reconstructed macroblocks within the same frame. For the luminance (luma) component, intra prediction uses 16x16 and 4x4 macroblock sizes. There are a total of 9 optional prediction modes for each 4x4 luma block and 4 optional modes for a 16x16 luma block. Due to the less sensitiveness and importance of chrominance components in an image, each chroma block is downsampled by a factor of two in both vertical and horizontal directions. Therefore, if the luma prediction uses 16x16 MB, corresponding chroma block size will be 8x8 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.

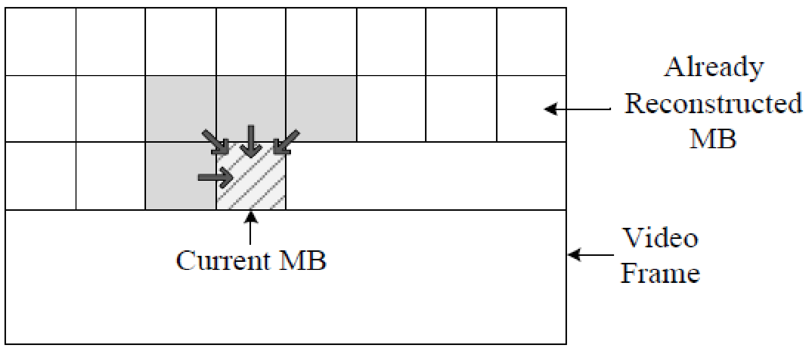


Figure 3.4: Intra Frame Prediction[9]

4x4 Luma Prediction Modes

There are nine prediction modes for 4x4 luma macroblock [9]. Figure 3.5 shows the luminance macroblock that is required to be predicted. The samples above and to the left have previously been encoded and reconstructed and are therefore available in the encoder and decoder to form a prediction reference. The prediction block P is calculated based on the samples labelled A-M in Figure 3.5. The arrows in Figure 3.6 indicate the direction of prediction in each mode.

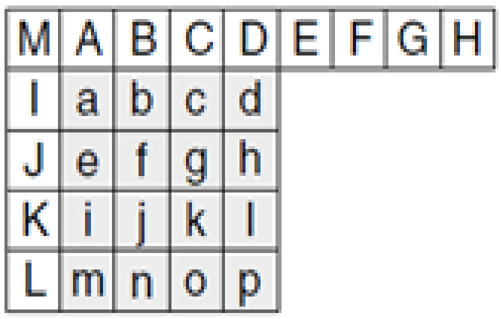


Figure 3.5: Labelling of prediction samples[9]

16x16 Luma Prediction Modes As an alternative to the 4x4 luma modes described in

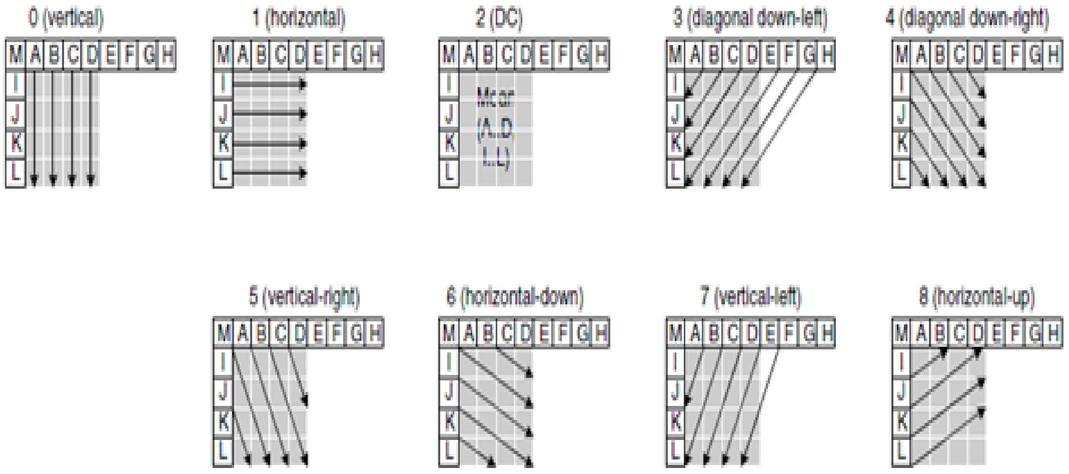


Figure 3.6: 4x4 Luma prediction modes[9]

the previous section, the entire 16x16 luma component of a macroblock may be predicted in one operation. There are 4 modes for 16x16 luma prediction mode [9]. Here the prediction is made from the horizontal and vertical pixels for all the modes.

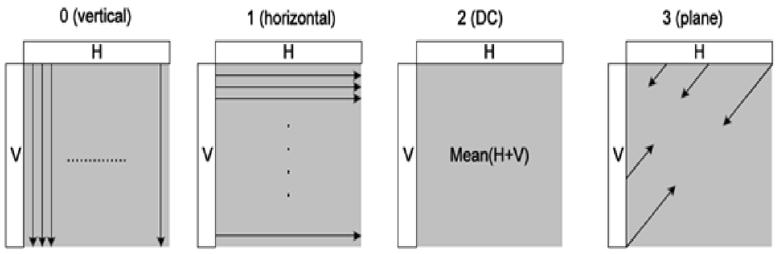


Figure 3.7: 16x16 Luma prediction modes[9]

8x8 Chroma Prediction Modes

Each 8x8 chroma component of a macroblock is predicted from chroma samples above and/or to the left that have previously been encoded and reconstructed. The 4 prediction modes are very similar to the 16x16 luma prediction modes described earlier and illustrated in Figure 3.7, except that the order of mode numbers is different: DC(mode 0), horizontal (mode 1), vertical (mode 2) and plane (mode 3) [9]. The same prediction mode is always applied to both chroma blocks.

3.3.2 Inter Frame Prediction

In inter prediction, the current frame is predicted from one or more previously encoded frames. The H.264/AVC encoder uses block-based motion compensation and supports variable block-size motion compensation ranges from 16x16 to 4x4 pixels. In inter prediction method, to predict a macroblock 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.

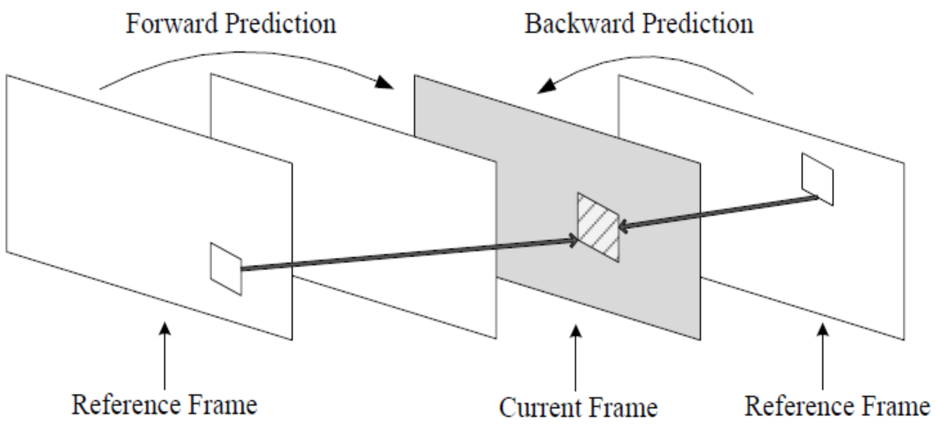


Figure 3.8: Inter Frame Prediction[9]

The luminance component of each macroblock (16x16 samples) may be split up in four ways (Figure 3.9) and motion compensated either as one 16x16 macroblock partition, two 16x8 partitions, two 8x16 partitions or four 8x8 partitions. If the 8x8 mode is chosen, each of the four 8x8 sub-macroblocks within the macroblock may be split in a further 4 ways (Figure 3.10), either as one 8x8 sub-macroblock partition, two 8x4 sub-macroblock partitions, two 4x8 sub-macroblock partitions or four 4x4 sub-macroblock partitions[9]. These partitions and sub-macroblock give rise to a large number of possible combinations within each macroblock. This method of partitioning macroblocks into motion compensated sub-blocks of varying size is known as tree structured motion compensation.

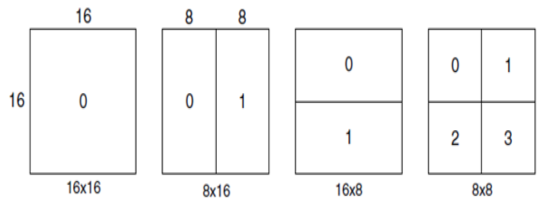


Figure 3.9: Macroblock partitions: 16 x 16, 8 x 16, 16 x 8, 8 x 8 [9]

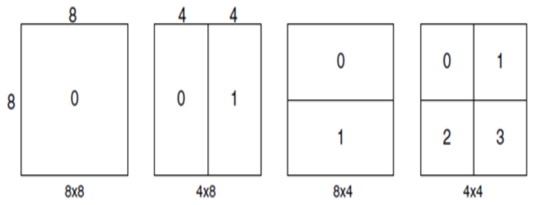


Figure 3.10: Sub-macroblock partitions: 8 x 8, 4 x 8, 8 x 4, 4 x 4 [9]

So, implementing the merging block in motion estimation loop requires merging to be implemented in all the above prediction modes.

3.4 Downsampling of Depth map

The main problem with the above proposed methodology is that all the bits of encoded depth shall not be merged in the encoded texture. The complete depth map is not accommodated in a single bitstream. The main reason for this is that opportunities of merging depth bits is lost due to skipped macroblock, 8x8 partition skip(luma and chroma), 4x4 block skip, luma Intra 16x16 DC, chroma DC and last level coefficient value is "1". One approach is proposed to accommodate the depth bits which are not merged by the technique shown in Figure 3.3. The approach deals with the downsampling of the depth map.

Downsampling prior to encoding have been used in the prior-art to improve the compression efficiency of multiview video coding with depth information [32], [24]. In MVVD coding, the depth map is encoded and not displayed. It is only used to synthesize intermediate views. So, better rendering quality rather than absolute depth quality is of prime importance. In [33], downsampling of an image to a low resolution was carried out followed by coding and subsequently interpolating the result to the original resolution leading in improvement of the overall performance of the compression process. Depth images do not contain any texture and are predominantly flat with sharp edges marking the boundary between objects at different depths. A reduced resolution depth compression method for multiview video plus depth coding is presented in [34] also. The B-spline down/up sampling methods were designed by considering the characteristics of the depth image and enormously reduced the depth bit-rate without degradation of rendering quality [34]. This motivated us to use such an approach in our current studies also.

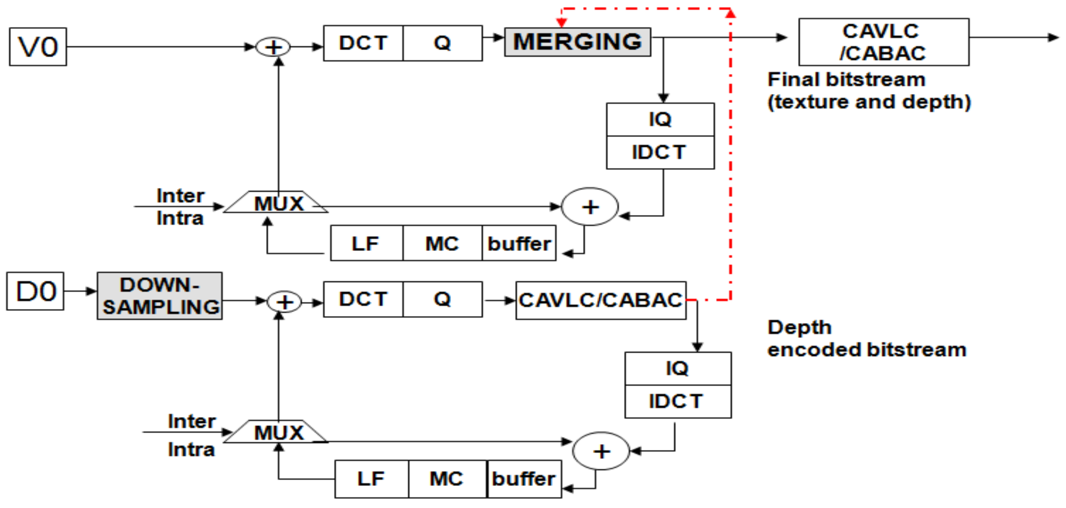


Figure 3.11: Downsampling as pre-processing block at depth encoding

A pre-processing block ”Downsampling” has been added at the encoder side. Figure 3.11 shows the ”Downsampling” block in the encoder phase. Basically, downsampling is applied in both horizontal as well as vertical direction. This reduces the depth encoded bits to around 23% to 27% if we downsample by factor 2. Here it is seen that depth map D0 is been downsampled and then coded before merging is applied. By reducing the number of depth bits, it is possible that the whole depth map is accommodated in the texture as a single bitstream.

Similarly, the merging of depth bits is implemented in encoder for multiple views. Our approach in encoder for multiple views is shown in the Figure 3.12. Here, texture view 1 (V1) is predicted from view 0 (V0) and depth view 1 (D1) is predicted from depth view 0 (D0). The bitstream of different views are generated and are given to the assembler to form the final bitstream containing of depth and texture.

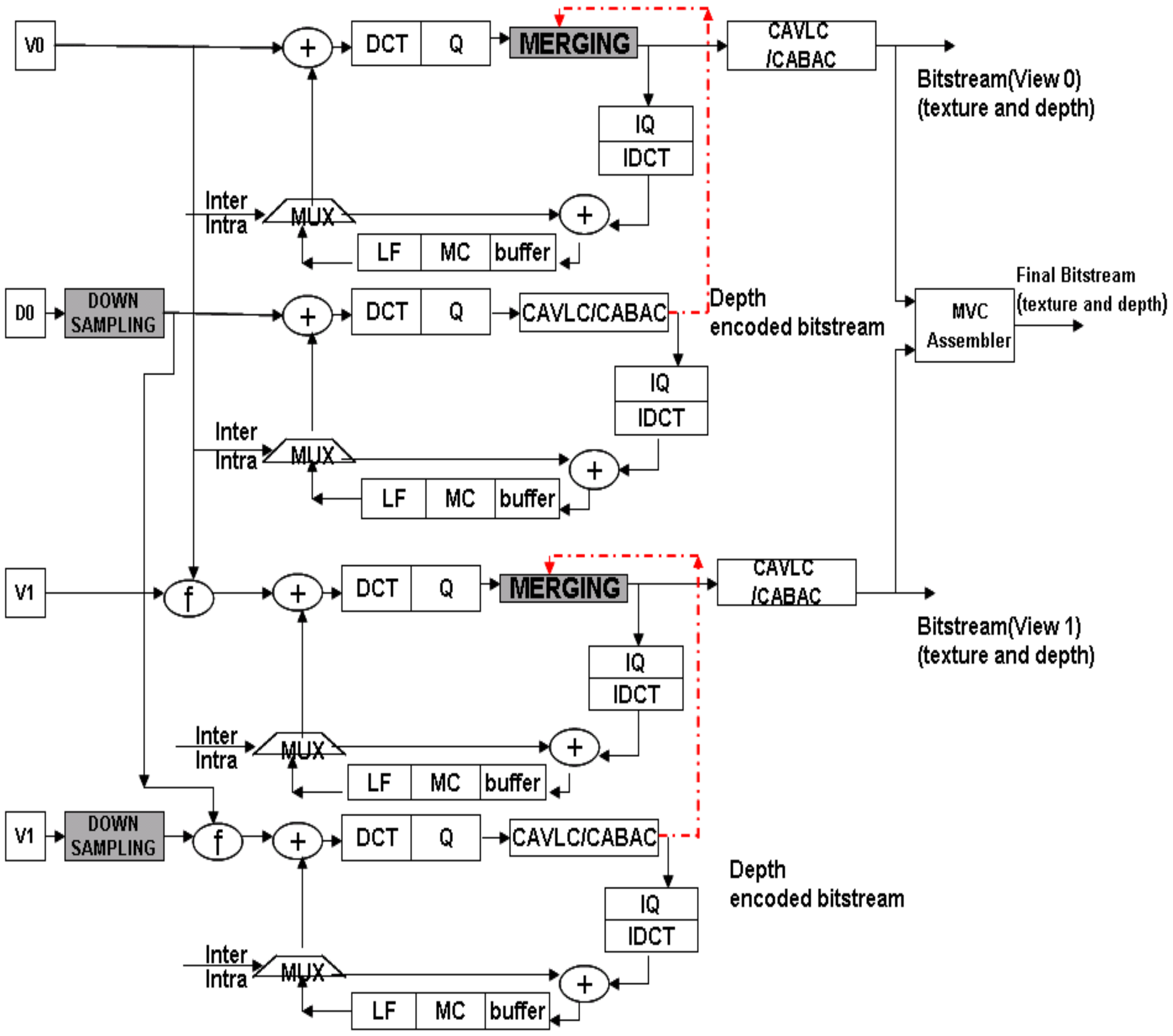


Figure 3.12: Proposed approach in Encoder for Multi-View

3.5 Signalling the decoder about the existence of depth

The encoding process should be designed in such a way that the generated encoded bitstream is able to be decoded by the simple decoder. JM decoder, AVC decoder, standalone JMVC decoder, etc should be able to decode the encoded bitstream. This shall make the encoding process backward compatible. Also it is necessary to signal the decoder about the existence of depth in the bitstream so that the smarter decoder(able to detect/decode depth) is able to detect the presence of depth in the bitstream and decode the depth map. It is also necessary to send the information about the resolution at which the depth is encoded. To make the bitstream backward compatible and to send the information about the existence of depth, the two reserved bits of SPS (Sequence Parameter Set) found from the H.264 ITU-T standard [35] are used. The normal decoder will discard these two bits while the smart decoder will consider the two bits and decode the bitstream.

Table 3.1: Signalling the decoder about the existence of depth

Reserved Bits of SPS		**Description**
0	0	No depth is present in the bitstream
0	1	Depth is present with the original resolution that of texture.
1	0	Depth is present with resolution divided by downsample factor 2.
1	1	Depth is present with resolution divided by downsample factor 4.

At the decoder side, the decoder should be modified in such a way that it will consider the two reserved bit and decode the depth based on the values of the two bits.

3.6 Appending the remaining depth bits

There may be some cases where all the depth bits are still not accommodated in respective encoded texture bitstream. So a new approach was devised in [4] at the network abstraction layer. The encoded bits, slice header onwards, have been merged inside the corresponding texture and the remaining depth bits are appended at the end of slice data of texture of each frame leading to a single bitstream. Here, it is assumed in the figure that there is only one slice per frame. Figure 3.13 shows the depth merged into texture using steaganography approach. Figure 3.14 shows the appending of the remaining unmerged bits to the texture at NAL.

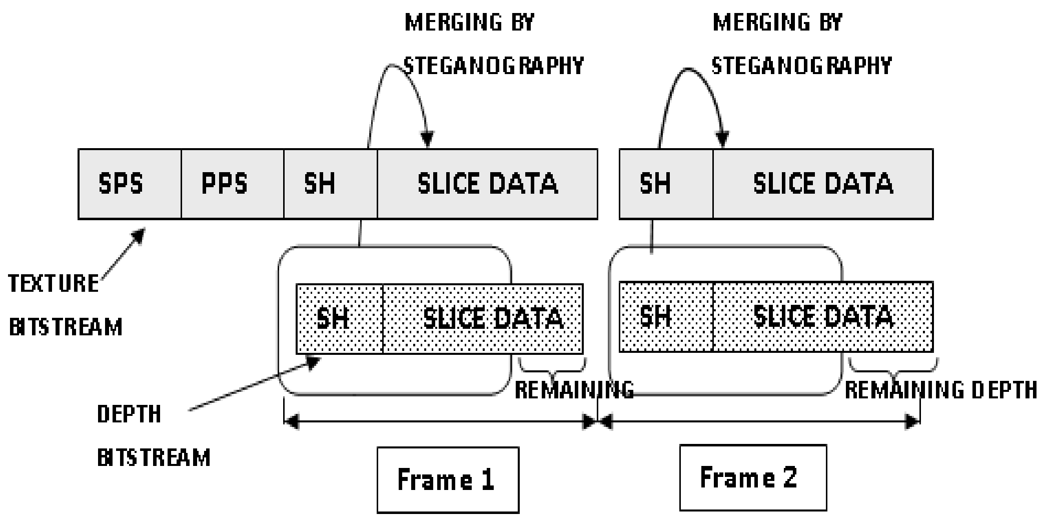


Figure 3.13: Merging of depth bits into texture [4]

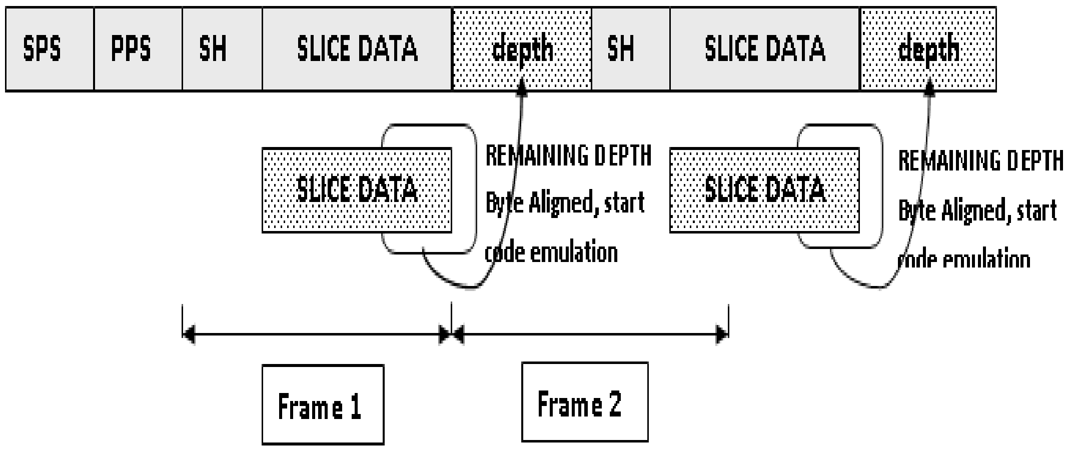

SPS
PPS
SH
SLICE DATA
depth
SH
SLICE DATA
depth
SLICE DATA
REMAINING DEPTH Byte Aligned, start code emulation
SLICE DATA
REMAINING DEPTH Byte Aligned, start code emulation
Frame 1
Frame 2

Figure 3.14: Appending remaining depth bits into the texture [4]

3.7 Decoding Phase

A smarter decoder is been developed which detects the presence of depth map in a bitstream, extracts the depth bits and decodes it.

Prior Implementation

The decoder for multiple view and depth is been implemented in [4]. At the decoder side, first the depth bits are extracted simultaneously alongwith the parsing of texture [4]. Then the texture is decoded followed by the decoding of depth [4].

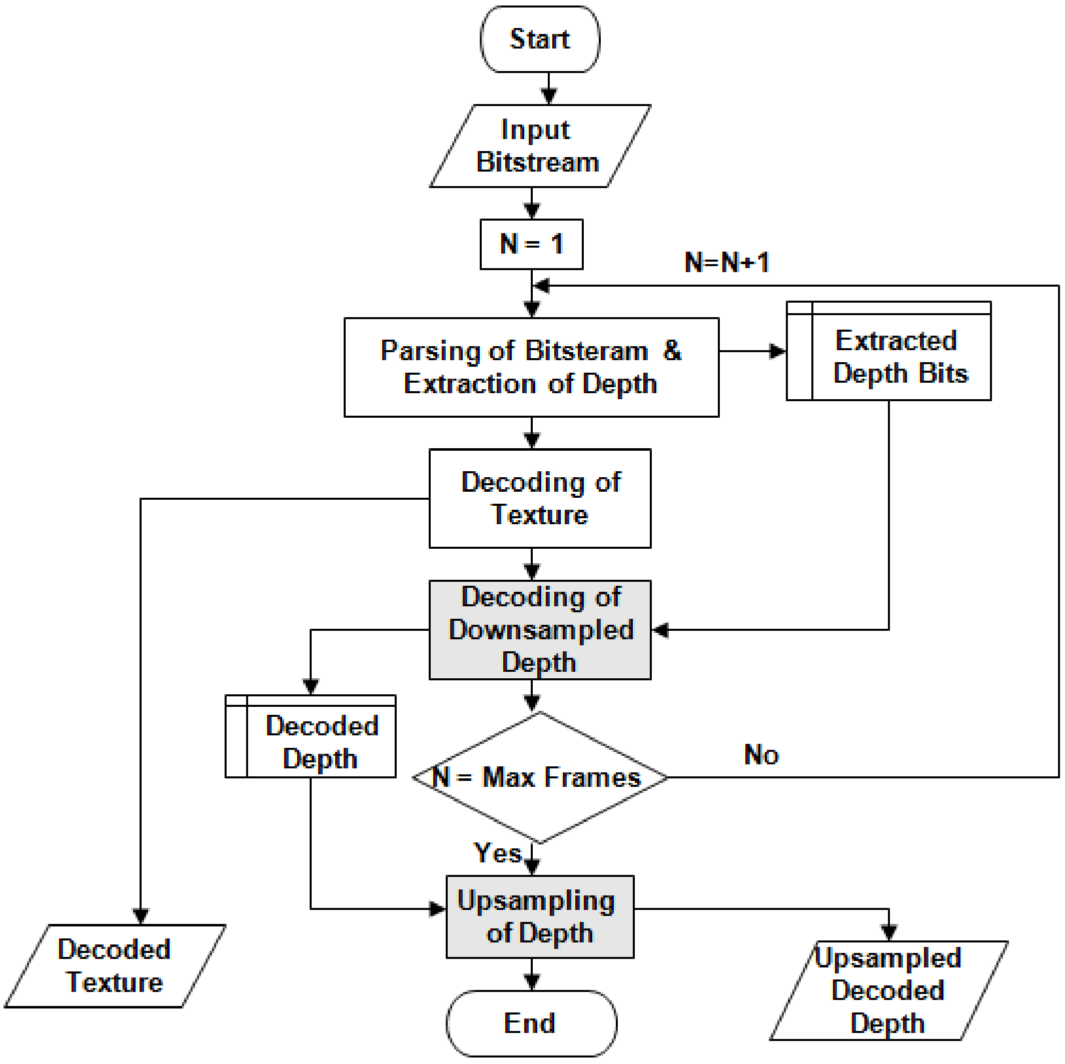


Figure 3.15: Flow Chart of Modified Decoder

Current Implementation

The decoder mentioned in [4] is not able to decode the depth which is encoded at different resolution i.e. downsampled depth. So, a modification was made in the decoder such that the decoder is able to decode the downsampled depth. After the depth decoding, the decoded depth is up-sampled to get the original resolution depth map. Fig. 3.14 shows the flow chart of the modified decoder. The core decoding procedure is same as that of H.264/MVC for multiple views. The complexity of basic decoder is the same as that of the decoder which encodes multiview video and depth as simulcast. Only an 'upsampling' block for depth map has been introduced as a post processing function.

3.8 Complexity of the entire encoding/decoding system

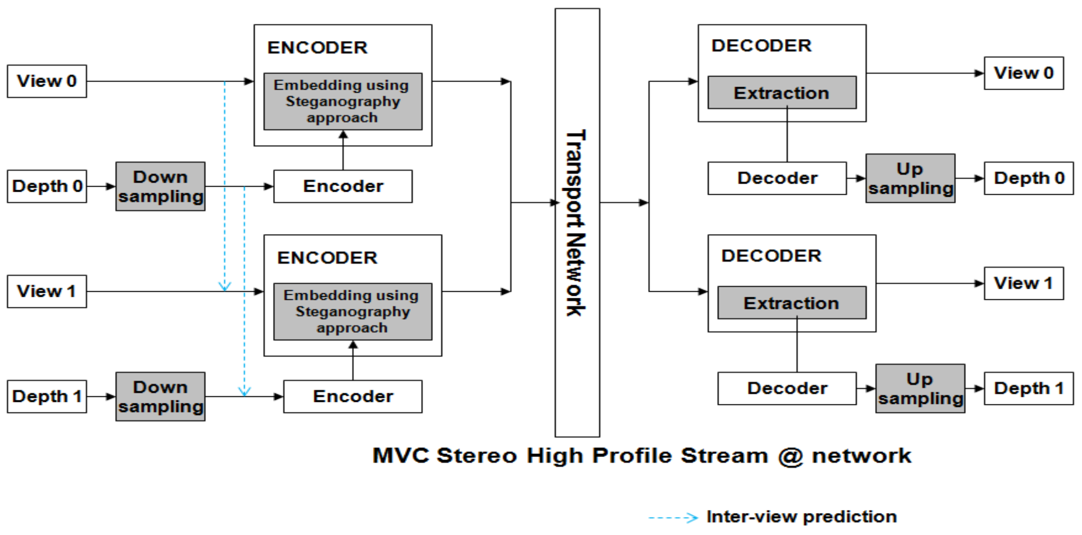


Figure 3.16: MVC Stereo High profile stream at Network

The above shows the complexity of the entire encoding-decoding system with the proposed methodology. The main advantage of the proposed methodology to the system is that no significant change is required with respect to the hardware. Only some blocks named Downsampling, Embedding using steganography approach, Extraction and Upsampling are introduced. So, this shall not make complex structure for the encoder as well as decoder block. Only some blocks are needed to be introduced in the entire coding chain.

Chapter 4

Implementation of the Proposed Techniques

This chapter deals with the implementation of proposed methodology of steganography for merging depth maps into texture information. For the implementation, we have used JMVC software which have been modified for implementing steganography for 3D video.

4.1 Tools and Software Used

The IDE tool used for the current implementation is Microsoft Visual Studio 2003. Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. Visual Studio supports different programming languages by means of language services, which allow the code editor and debugger to support nearly any programming language. The version used for the implementation is jmvc8.0. The entire code is in CPP.

JMVC (Joint Multi-view Video Coding) jmvc8.0, software is open source software for the Multi-view Video Coding (MVC) project of the Joint Video Team (JVT) [36] of the ISO/IEC Moving Pictures Experts Group (MPEG) and the ITU-T Video Coding Experts Group (VCEG). The JMVC software is written in CPP and is provided as source code. This is basically used to encode and decode only the texture views of the video frames as the input. So some modifications had been done to the JMVC.

4.2 Earlier implementations in JMVC (jmvc8.0)

Some new implementations have been done in JMVC(jmvc8.0) code by modifying the source code. The implementations which were done earlier are as under:

- The original jmvc8.0 code was for encoding and decoding only texture views of a video as input. This code was modified in such a way that it could encode and decode monochrome depth maps of a video as an input.
- The above code was extended for multiple views and associated depth maps in such a way that we get the two separate bitstreams for texture and depth for multiple views.
- Multipass encoder was implemented in which depth and texture is encoded in each pass.

But in these implementations, channel bandwidth is directly proportional to number of views and depth maps as with MVC. Thus 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 by implementing the proposed methodology.

4.3 Current implementations in JMVC (jmvc8.0)

For implementing the concept of merging depth maps into texture maps, some changes must be made in the above code for jmvc8.0. For implementing the merging concept, we need to have the depth encoded bitstream for each frame so that we can merge this encoded depth bits into the texture encoded bits for the respective frames.
So the entire task is divided into following sub-tasks:

- **Encoding of texture (view) and depth map:**As discussed earlier, in one pass monochrome depth-map is encoded for a frame and in second pass texture view is encoded for the corresponding frame, so two passes for one frame, and this continues to go on for other frames also. Also two different reconstructed videos (recon_colour_V.yuv, recon_depth_V.yuv) one for texture and another for depth where V is the view number are created.

- **Dumping:** Now to merge the encoded depth frame bits in texture frame bits these depth bits should be stored 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 (texture frame).

- **Merging:** The data which has been dumped in dump_X.bin (where X is the frame number) is merged with the encoded bits of texture at the macro-block level. During merging some of the following points need to be considered

Merging concept is applied at each transform 4x4 blocks, except in the blocks of Intra_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 (depth merged into texture), merging is avoided when

- Macroblock is skipped.

- 8x8 partition is skipped based on coded block pattern (cbp) value.

- Number of significant coefficients is zero. That is the merging should be done only if there is atleast one non-zero coefficient in the 4x4 block.

- Intra_16x16_Luma_DC coefficients.

- Chroma_DC coefficients.

- If the value of the last significant coefficient is 1, then the merging is avoided here to remove the ambiguity by the decoder. If merging is applied and after merging the value of the last level is 1, the decoder fails to detect whether there is merged bit or not.

The bitstream we get after merging is processed as usual through CABAC entropy coding. After the entropy coding the final encoded bit stream obtained is consistent with 2D decoder.

4.4 Development of add/remove black line Tool

There are many configurations where the sequences are encoded at different resolutions. The frame width and height should be in such a way that they should be multiple of 16 because the basic size of the macroblock is 16x16. When we apply downsampling to the sequences, there may be a case where after downsampling the frame by factor 2 the width and height of the frame might not be in multiple of 16. For example, a sequence with 720p resolution. It is necessary to convert to a required resolution which is a multiple of 16. So, a black line is added at the end for texture as well as monochrome video. Also the removal of black line is also implemented for the same so that the original video is regenerated.

Chapter 5

Simulation Results and Analysis

This chapter deals with the simulation results and the analysis derived from it by implementing the merging concept using the algorithm of steganography. Simulations are carried out on different sequences with each and every possible configuration. Also the different sequences with different resolution are used for simulation results. The simulations are carried out for all the above proposed methodologies. Different QP values have been used for encoding the depth maps so that we have the flexibility to adjust the number of bits and quality of depth maps. To improve the quality of depth maps we also used same QP values to encode both textures and depth maps.

5.1 Comparison for the simulation results for merging in final bitstream with merging in motion estimation loop

Simulations on different sequences with CIF(352x288) resolution are carried out. The input stream contains 15 frames and two texture views with its associated depth maps. Texture views contain luminance and chrominance components while depth maps contain only luminance components. Different QP values for texture and depth have been used because change in a value of QP generates different number of bits in an encoded bit-stream thereby capturing all possible effect on quality. First, we applied merging directly at the final bit-stream as shown in Figure 3.1. Two configurations have been explored for comparison purposes. In one configuration, two views of texture alongwith associated depth maps have been encoded into a single bitstream consisting of depth and texture as shown in Figure 3.1. This is termed as ”Merging in bitstream”. For benchmarking we configured the encoder as conventional H.264/MVC approach. In this configuration, two views of textures alongwith two views of associated depth maps have been encoded into single bitstream consisting of 4 views which is termed as ’Anchor’ hereafter. Both the configurations have been simulated for high delay applications i.e. I, P and B pictures. MSR_BreakDancers CIF (352x288) sequence is used as an input stream for the simulations. The texture encoding is done at Quantization Parameters (QPs) = 14, 18, 22 and 26 and the depth encoding is done at QPs = 32, 36, 40 and 44. The GOP = 8, Intra Period = 16 and Number of Frames = 15.

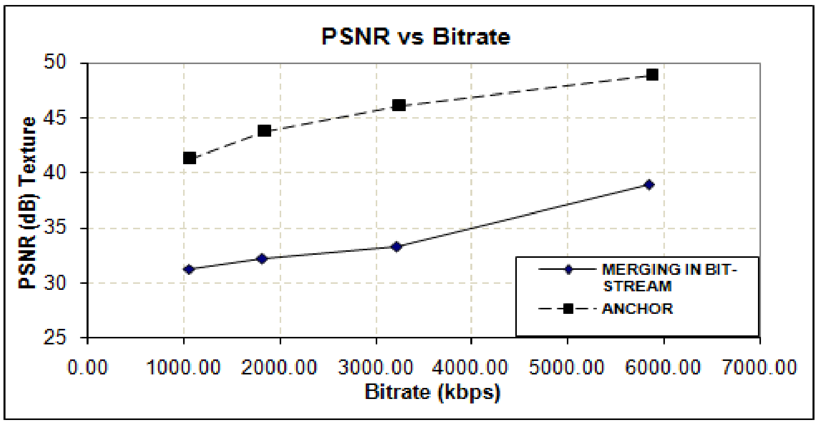


Figure 5.1: Merging in final bitstream for MSR_Breakdancer

Table 5.1: Merging statistics of MSR_Breakdancer sequence

Sequence: MSR_BreakDancer, GOP=8, Intra period=16			
QPtexture, QPdepth	**Depth bits to be merged**	**Depth bits Merged**	**% Depth bits merged**
14 , 32	163104	119995	**73.57**
18 , 36	95984	71890	**74.90**
22 , 40	55856	42262	**75.66**
26 , 44	33544	22833	**68.07**

From the above graph, we can derive that there is a significant drop in PSNR of texture of around 8 dB to 10 dB compared to that of ”Anchor” and merging percentage of depth is around 73%. Thus the quality of picture is degraded in a visible manner which is unaccept-

able. Reason behind this degradation is that motion estimation loop at the encoder doesn't compensate for the degradation in the final bitstream since the encoder doesn't have any feedback for the modification at the final bitstream. So at the decoder side the error keeps on propagating which in turn results to a high loss in PSNR. To avoid this problem we implemented merging inside the motion estimation (ME) loop as in Figure 3.3.
Now, keeping all the parameters and configuration same as above, simulation is carried out on the configuration where merging is applied in motion estimation loop. This is termed as "Merging in motion estimation".

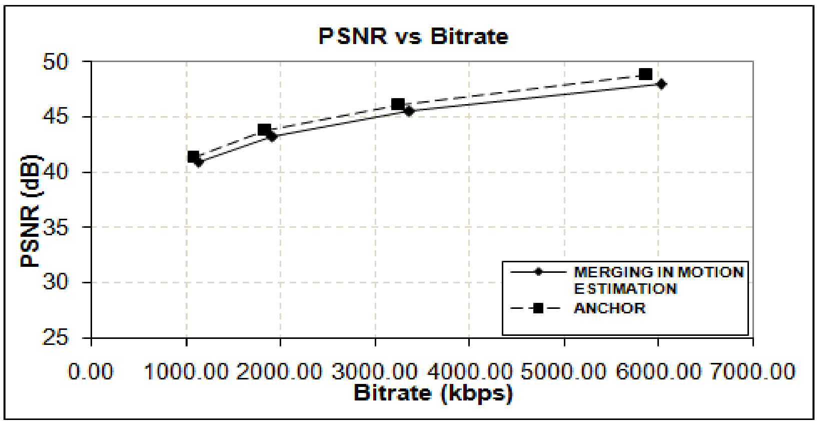


Figure 5.2: Merging in motion estimation for MSR_Breakdancer

Table 5.2: Merging statistics of MSR_Breakdancer sequence

Sequence: MSR_BreakDancer, GOP=8, Intra period=16			
QPtexture, QPdepth	**Depth bits to be merged**	**Depth bits Merged**	**% Depth bits merged**
14 , 32	163104	107933	**66.17**
18 , 36	95984	66087	**68.85**
22 , 40	55856	38445	**68.83**
26 , 44	33544	20548	**61.26**

Thus, when merging is implemented in motion estimation loop, the degradation in quality of texture is around 0.4dB to 0.6dB which is acceptable. Also, there is not much effect on the amount of merging. So, now onwards we shall apply the merging concept in motion estimation for the simulations.

5.2 Simulation Results with merging in only 4x4 mode

In this section, simulations were carried out on different HD sequences, 100 frames with merging in only 4x4 transform block. For the current simulations, 8x8 transform block is disabled. Two configurations are considered. One configuration is 'Anchor' while second configuration is 'Downsample + Merging at last level'. Currently, simulations are carried out for low delay applications (with B pictures). Figure 5.3 and 5.4 shows the R-D curve for MSR_Breakdancers and GT_Fly respectively. Merging statistics for the same are shown in Table 5.3 and 5.4.

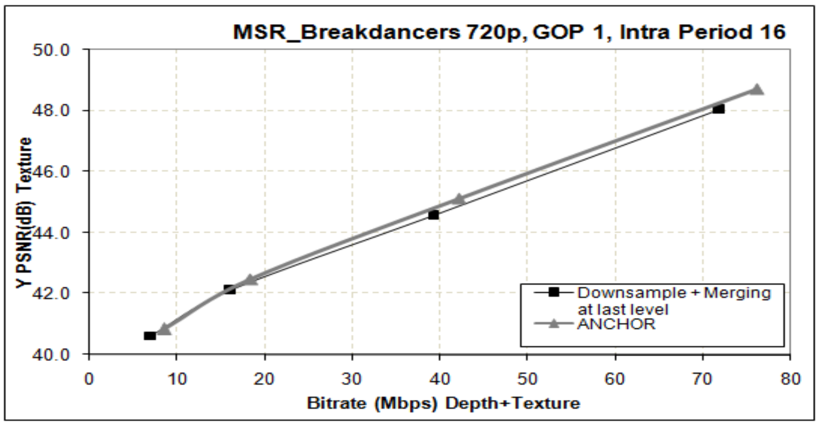


Figure 5.3: R-D performance for MSR_Breakdancers Low Delay (Merging in 4x4)

Table 5.3: Merging statistics of MSR_Breakdancers sequence (Downsample + Merging in 4x4)

Sequence: MSR_BreakDancer, GOP=1, Intra period=16			
QPtexture, QPdepth	**Depth bits to be merged**	**Depth bits Merged**	**% Depth bits merged**
14 , 14	14916536	6438760	**43.17**
18 , 18	10035280	3986137	**39.72**
22 , 22	6697712	1568416	**23.42**
26 , 26	4293056	591660	**13.78**

Simulations were also carried out for high delay applications which shows the same characteristics. From the above R-D curves it can be seen that the loss in PSNR of texture

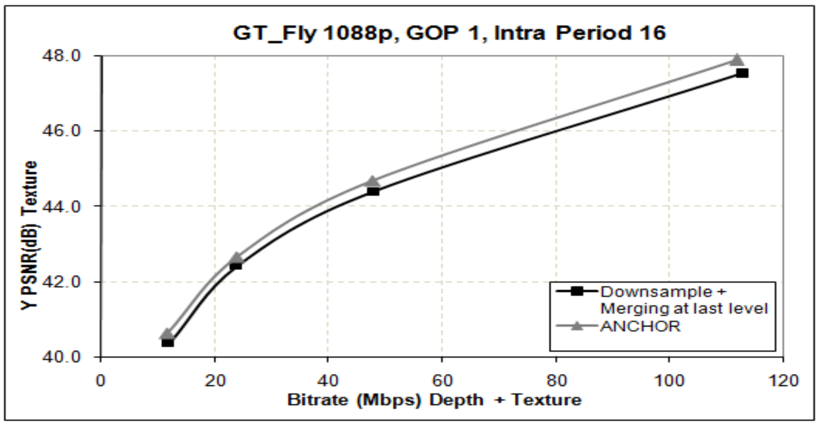


Figure 5.4: R-D performance for GT_Fly Low Delay (Merging in 4x4)

Table 5.4: Merging statistics of GT_Fly sequence (Downsample + Merging in 4x4)

Sequence: GT_Fly, GOP=1, Intra period=16			
QPtexture, QPdepth	**Depth bits to be merged**	**Depth bits Merged**	**% Depth bits merged**
14 , 14	6663472	6340672	**95.16**
18 , 18	3846624	3492868	**90.80**
22 , 22	2198128	1943285	**88.41**
26 , 26	1344344	1072232	**79.76**

compared to Anchor is around 0.6dB and the bitrate is also reduced by around 4% to 6%. The amount of merging of depth bits is good when merging is applied only in 4x4 block.

5.3 Simulation Results of Final Proposed Methodology

In this section, simulation results carried out on different sequences are presented and the relation among each and every approaches proposed discussed in chapter 3 are shown. Different HD streams, both 768p and 1080p with different range of QP values and 100 frames in a sequence are used to carry out simulations for two texture views and associated two depth maps. Also two different types of sequences are used, natural sequences and synthetic sequences.

Following configurations are used for the simulations which are carried out:

- Encoded two views and its associated depth maps.
- 8x8 Transform mode enabled by enabling FREXT mode.
- Both High Delay (I,P,B frames) and Low Delay (I,P frames) applications.
- Number of frames is 100.
- " QP(texture) = QP(depth) = 22, 26, 30 and 34. The values of QPs are selected to achieve the bitrates suggested as per values given in MPEG2011/N12026 3DV-Cfp[17].

Following sequences used for the simulations are:

- Natural sequences (Kendo, Nagoya_Balloons, GIST_Newspapers). All at resolution 768p.
- Synthetic sequences (Nokia_Dancer, Nokia_GT_Fly). All at resolution 1080p.

Three configurations have been explored for comparison purposes. In one configuration, two views of textures alongwith two views of associated depth maps have been encoded into single bitstream consisting of 4 views which is termed as 'Anchor' hereafter. In second configuration, two views of depth maps have been merged into the bitstream of associated textures as shown in Figure 3.3, termed as 'Merging at last level'. In the third configuration, the depth map is downsampled by factor 2 (Figure 3.12) and the remaining depth bits (if left after merging) are also appended at network abstraction layer as described in chapter 3 (Figure 3.14). This is termed as 'Downsampled + Merging at last level' hereafter. Thus, all the depth bits (merged and not merged) are considered in this case and a single bitstream containing entire texture and entire depth map is generated.

Simulation results shown below represent the whole system consisting of two views and two associated depth maps and the merging statistics. Currently, the simulation results are carried out for low delay applications i.e. without B pictures. The performance of the current study in terms of compression efficiency for the natural sequence 'Kendo' sequence is shown in Figure 5.5 and merging statistics is shown in Table 5.5.

From the R-D curve as shown in Figure 5.5, for 'Merging at last level' there is a negligible degradation in PSNR of texture of around 0.3 dB compared to 'Anchor' and there is an

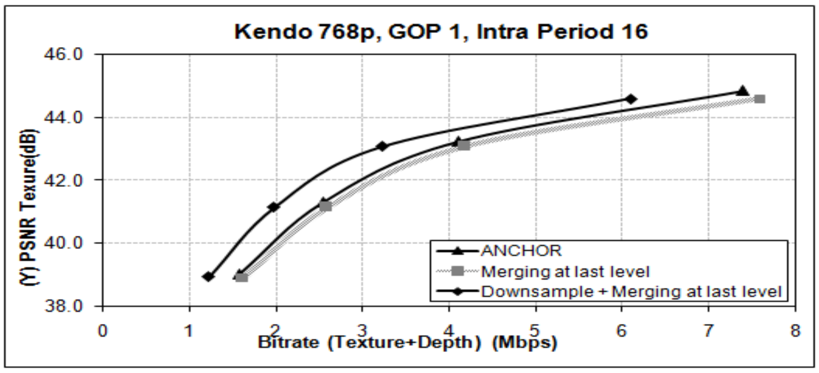


Figure 5.5: R-D performance for Kendo (Low Delay))

Table 5.5: Merging statistics of Kendo sequence for Low Delay (Downsample + Merging)

Sequence: Kendo, GOP=1, Intra period=16			
QPtexture, QPdepth	**Depth bits to be merged**	**Depth bits Merged**	**% Depth bits merged**
22 , 22	4371176	349340	**7.99**
26 , 26	2748928	162605	**5.92**
30 , 30	1718184	81605	**4.75**
34 , 34	1039760	40959	**3.94**

increase in bitrate of around 2% to 3%. For 'Downsample + Merging at last level' there is a considerable decrease in bitrate of around 15% compared to 'Anchor'.

The above simulations were also carried out for other two natural sequences. Figure 5.6 and 5.7 shows the R-D curves for 'Balloons' and 'GIST_Newspapers' respectively. As there is no significant change in merging amount, the merging statistics are not been presented.

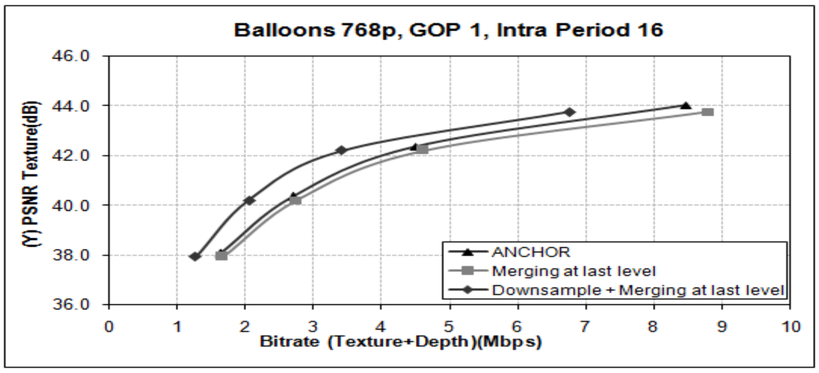


Figure 5.6: R-D performance for Balloons (Low Delay)

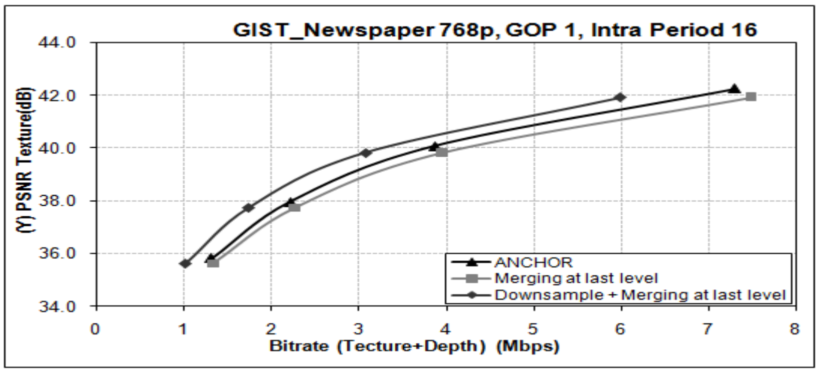


Figure 5.7: R-D performance for Balloons (Low Delay)

Further, the above simulation was carried out for synthetic sequence GT_Fly. Figure 5.8 shows the R-D curve and Table 5.6 shows the merging statistics.

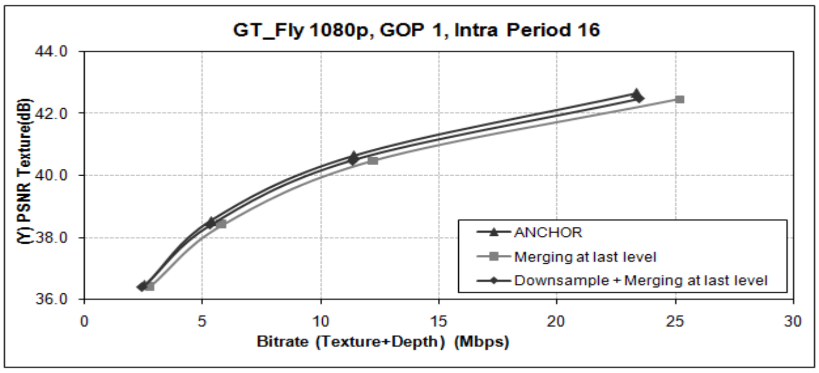


Figure 5.8: R-D performance for GT_Fly (Low Delay)

Table 5.6: Merging statistics of GT_Fly sequence for Low Delay (Downsample + Merging)

Sequence: GT_Fly, GOP=1, Intra period=16			
QPtexture, QPdepth	**Depth bits to be merged**	**Depth bits Merged**	**% Depth bits merged**
22 , 22	2108120	1419962	**67.36**
26 , 26	1299992	690869	**53.14**
30 , 30	831552	303313	**36.48**
34 , 34	525648	108662	**20.67**

From the R-D curve as shown in Figure 5.9, for 'Merging at last level' there is a negligible degradation in PSNR of texture of around 0.2 dB compared to 'Anchor' and there is an increase in bitrate of around 2% to 3%. For 'Downsample + Merging at last level' there is a negligible increase in bitrate to 'Anchor'. It can be seen that for synthetic sequence, even after downsampling of depth map there is no decrease in bitrate compared to 'Anchor'. The main reason for this is that the actual overhead of the encoded depth is around 2% to 5% compared to encoded texture compared to that for natural sequences which is around 25% to 45%.

The above simulations were also carried out for one more synthetic sequence. Figure 5.9 shows the R-D curve for 'Nokia_Dancer'. As there is no significant change in merging amount, the merging statistics are not been presented.

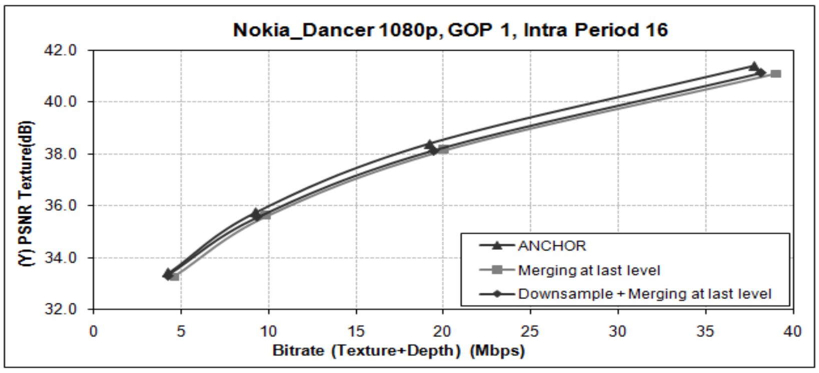


Figure 5.9: R-D performance for Nokia_Dancer (Low Delay)

Similarly, the statistics for High Delay applications i.e. with B pictures are generated. Here, for high delay applications, only the amount of merging is decreased due to presence of B pictures. Figure 5.10, 5.11, 5.12, 5.13, 5.14 shows the R-D performance of the entire system for sequences Kendo, Balloons, GIST_Newspapers, Nokia_Dancers and GT_Fly respectively. Table 5.7 and Table 5.8 shows the merging statistics for Kendo and GT_Fly respectively.

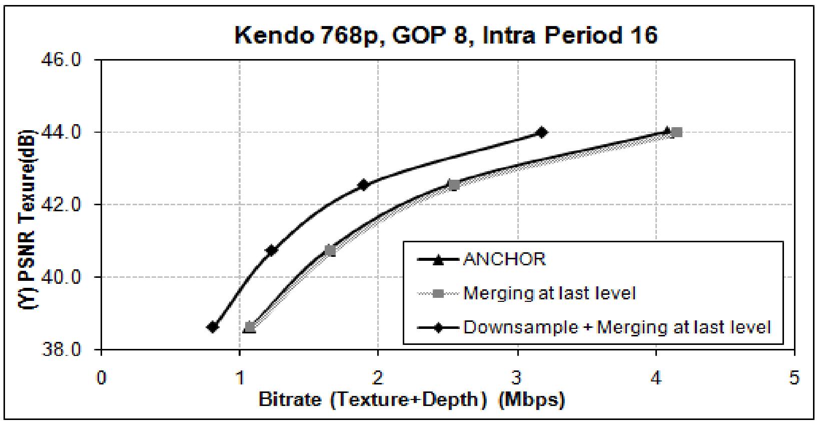


Figure 5.10: R-D performance for Kendo (High Delay)

Table 5.7: Merging statistics of Kendo sequence for High Delay (Downsample + Merging)

Sequence: Kendo, GOP=8, Intra period=16			
QPtexture, QPdepth	**Depth bits to be merged**	**Depth bits Merged**	**% Depth bits merged**
22 , 22	2639352	155498	**5.89**
26 , 26	1696400	79169	**4.67**
30 , 30	1080168	42950	**3.98**
34 , 34	675216	23071	**3.42**

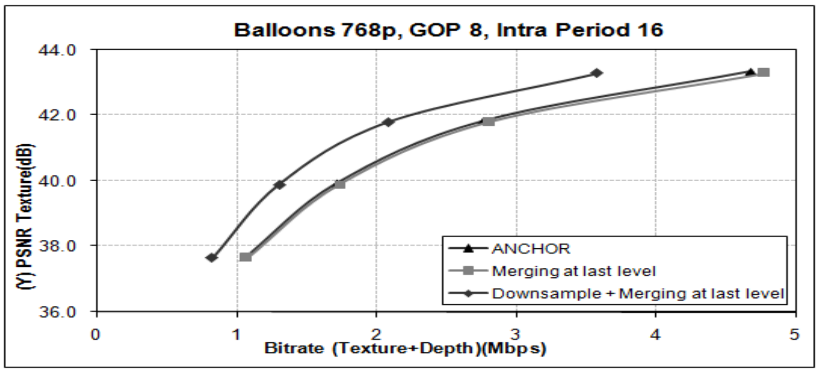


Figure 5.11: R-D performance for Balloons (High Delay)

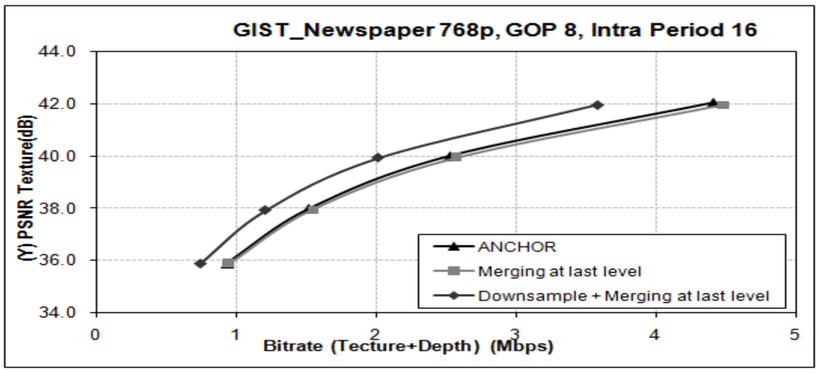


Figure 5.12: R-D performance for GIST_Newspaper (High Delay)

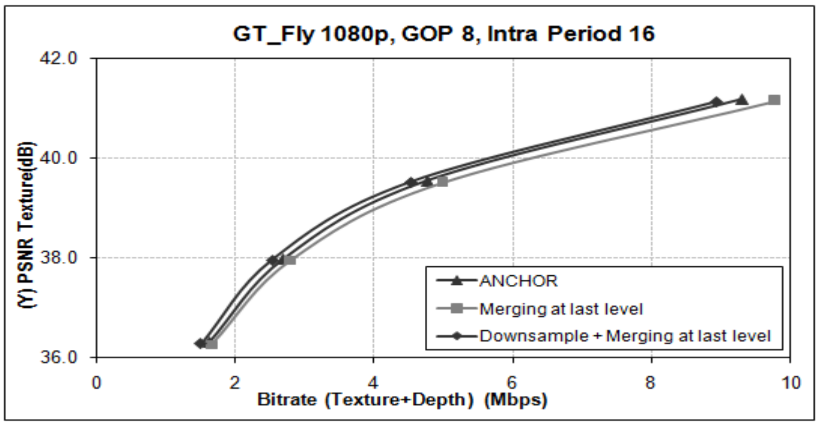


Figure 5.13: R-D performance for GT_Fly (High Delay)

Table 5.8: Merging statistics of GT_Fly sequence for High Delay (Downsample + Merging)

Sequence: GT_Fly, GOP=8, Intra period=16			
QPtexture, QPdepth	**Depth bits to be merged**	**Depth bits Merged**	**% Depth bits merged**
22 , 22	1131048	432685	**38.26**
26 , 26	702208	188952	**26.91**
30 , 30	456824	92343	**20.21**
34 , 34	279776	47770	**17.07**

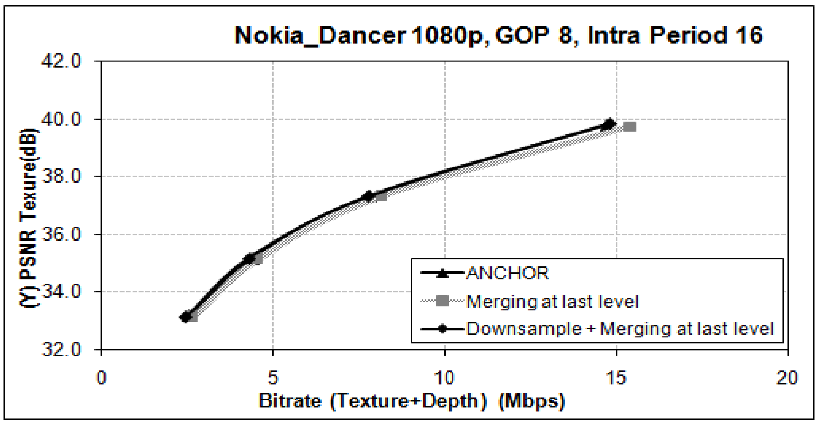


Figure 5.14: R-D performance for Nokia_Dancer (High Delay)

If we consider only 4x4 transform block, then ideally 24 bits of depth should be merged in one macro-block of texture, 16 bits into 16 luma blocks and 8 bits into chroma blocks. When we analyze these statistics at each macroblock level, it is seen that the same amount of bits are not merged due to: a) no merging due to skipped blocks because of invalid coded block pattern, b) no merging can take place when number of significant coefficient is equal to zero, which means no non zero coefficient present in the block, c) No merging have been done when the value of last level coefficient is "1". This is very essential so that a standard H.264/MVC decoder is able to decode two texture views seamlessly.
It has been observed that the bitrate for 'Downsampled + Merging at last level' has considerably decreased (16%) compared to that of 'Anchor'. Also the degradation in PSNR is upto maximum 0.3 dB. All these results show that the proposed methodology works best for natural sequences (Kendo, Balloons and GIST_Newspaper) as the bitrate is reduced considerably without affecting much on the quality of texture. Not much improvement has been observed for synthetic sequences (GT_Fly). The main reason for this is that the overhead of depth map is actually 2% to 5% to that of texture compared to 15% to 45% for natural sequences. So due to less depth overhead the advantage by the studied methodology is not visible.

5.4 Effect of Downsampling

As described earlier, depth maps are first downsampled and then encoded. At the decoder side the reverse is applied i.e. first decoded and then upsampled. Because of this, there is a considerable drop in PSNR of depth which is maximum 8 (natural scene) - 14dB (synthetic scene). PSNR is not always very conclusive especially when we consider the subjective quality, so we used SSIM metric also for measuring the quality of depth map alone. SSIM is devides based on the assumption that human visual perception is highly adapted for extracting structural information from a scene. Basically, SSIM is been calculated based on the compbinations of three components: luminance measurement, contrast measurement and structure comparison [37]. Figure 5.15 and 5.17 shows the effect of downsampling/upsampling on the PSNR of depth map for Kendo and Nokia_Dancer respectively and Figure 5.16 and Figure 5.18 shows the SSIM results for the same.

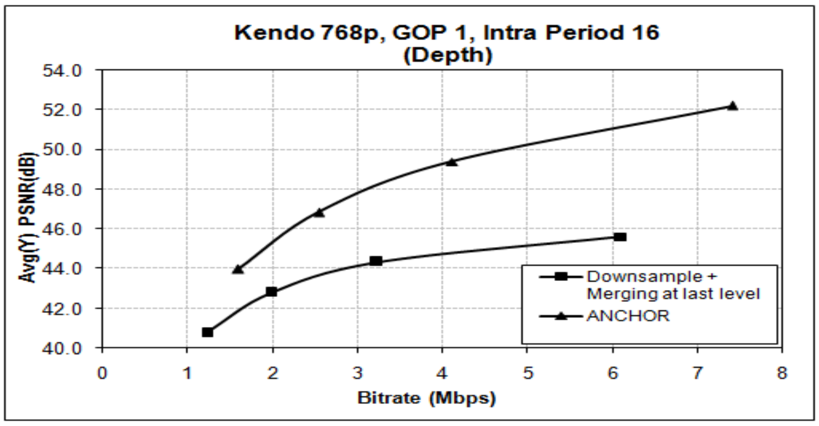


Figure 5.15: Bitrate vs PSNR (Depth) for Kendo

Similar experiments on other sequences also reveal that the SSIM results are quite encouraging even though the PSNR is quite unacceptable.

Since the ultimate quality of any 3DV/FVV scheme is largely dependent on the quality of the rendered views on the final display, we evaluated the quality of the synthesized views at the centre of the encoded/decoded views for illustration. View Synthesis was carried out for synthesizing the intermediate views from the given two views. The software used for the view synthesis is FVVSynthesis4.0. For example, we have generated the synthesized video

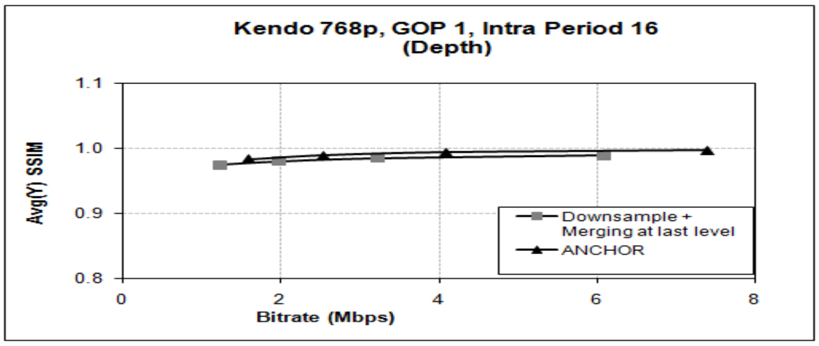


Figure 5.16: Bitrate vs SSIM (Depth) for Kendo

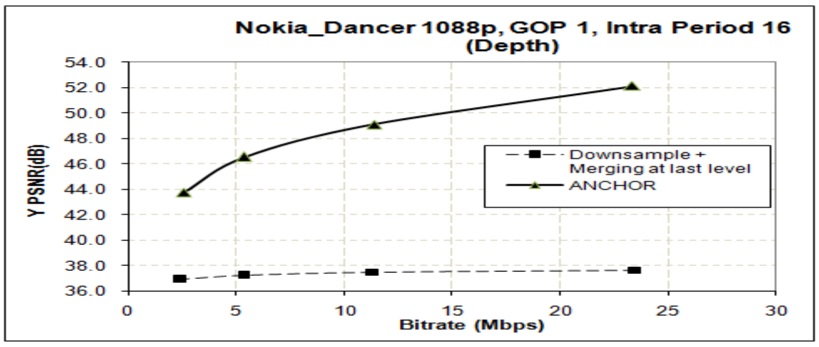


Figure 5.17: Bitrate vs PSNR (Depth) for Nokia_Dancer

at view 4 from views 3 and 5 and associated depth maps as per the current studies. Figure 5.19 shows snapshot of the synthesized view for Kendo sequence with the configuration 'Merging at last level without downsampling/upsampling of depth map' and Figure 5.20 shows the same with downsampled/upsampled depth maps. Both these configurations have been compared with the reference view 4 which also available as reference.

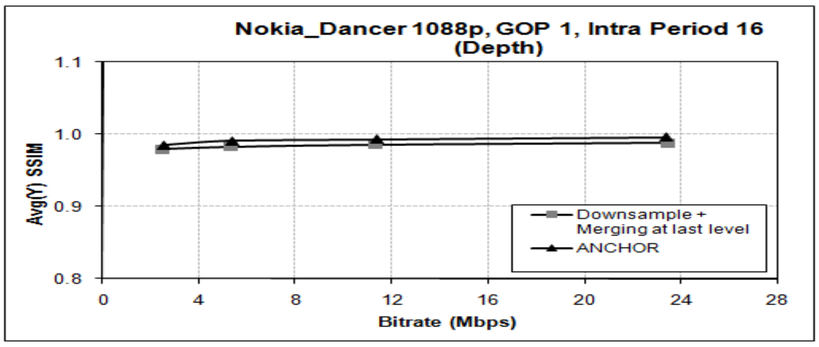


Figure 5.18: Bitrate vs SSIM (Depth) for Nokia_Dancer

Since depth maps are only used for virtual view rendering as non visual data, the perceived quality of the virtual views which are rendered using the decoded texture and depth maps is more important than the visual quality of depth map. Though the quality of depth is degraded in terms of PSNR, the visual quality of the synthesized view does not show the quality degradation. Also, the bitrate v/s SSIM graph for the above synthesized configuration, shown in Figure 5.21 suggests that the quality of the synthesized view is affected negligibly by the loss in quality of downsampled/upsampled depth.
As confirmation, experiments have been carried out in other sequences also with different configurations.

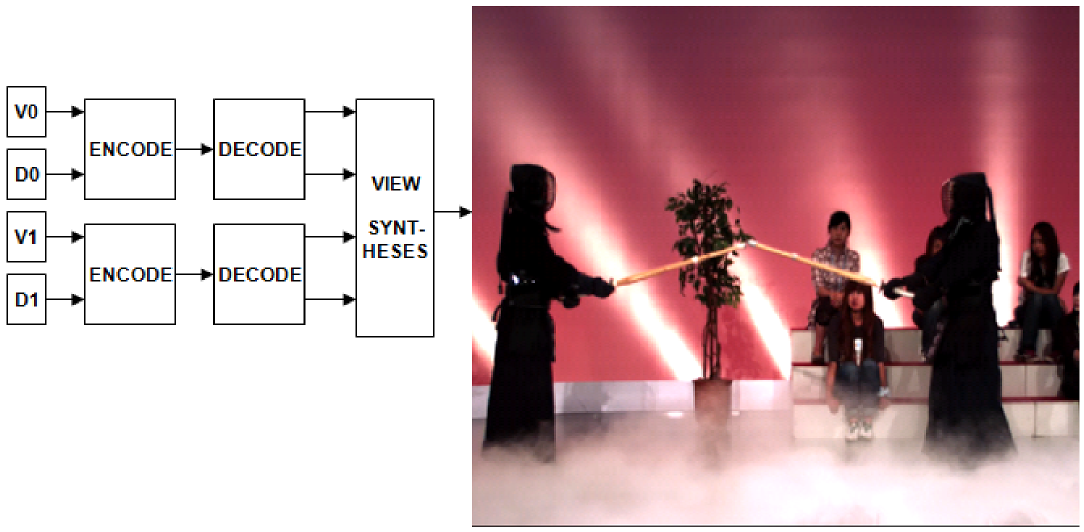


Figure 5.19: View synthesis results for Kendo with no downsampling/upsampling of depth maps and 1-bit merging

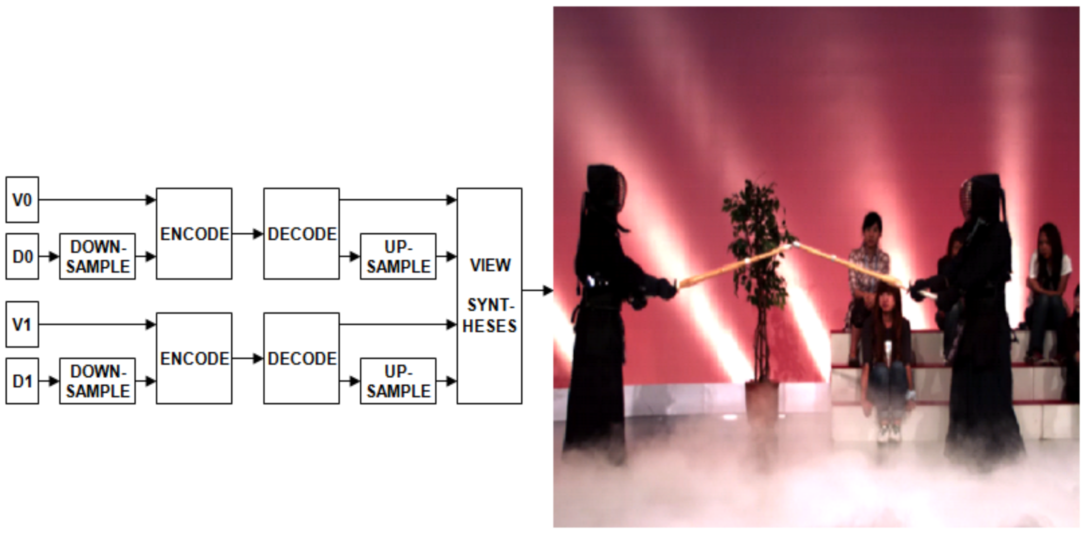


Figure 5.20: View synthesis results for Kendo with downsampling/upsampling of depth maps and 1-bit merging

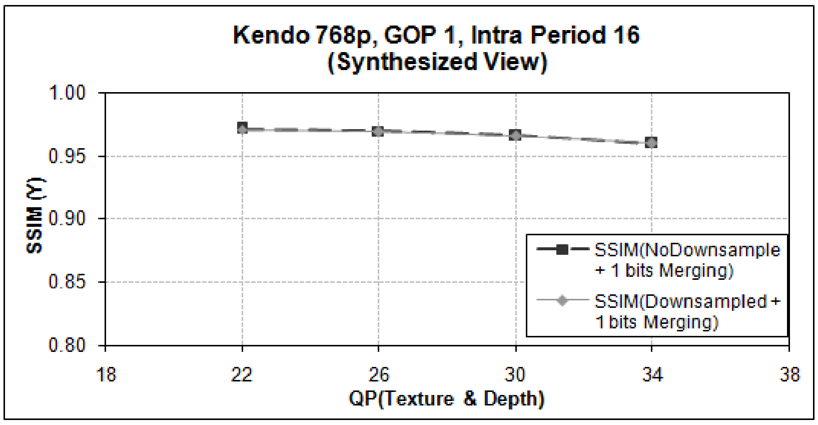


Figure 5.21: QP vs SSIM for View Synthesis for Kendo

5.5 Analysis

From all the above simulation results, both texture as well as depth is encoded at the various range of QP, the results shows less amount of merging of encoded depth bits in the texture frame. The reason for this is discussed above i.e.

- No merging due to skipped blocks because of invalid coded block pattern.
- No merging can take place when number of significant coefficient is equal to zero, which means no non zero coefficient present in the block.
- No merging has been done when after merging the last level coefficient value of a block is "1".

This is very essential so that a standard H.264/MVC decoder is able to decode two texture views seamlessly.
Following are the various points that are derived from all the above simulation results:

- Merging in Motion Estimation has improved the quality of the texture compared to merging in final bitstream.
- Merging of depth bits lead to negligible decrease in PSNR of around 0.2dB for natural sequences and 0.2dB for synthetic sequences.
- Merging of depth bits lead to negligible increase in bitrate of around 2% to 3% for natural sequences and 7% to 8% for synthetic sequences. The main reason for this is that error is introduced in the residual data of texture by merging the depth bits.
- Applying downsampling as a pre-processing block to the depth encoding, the original amount of encoded source depth is reduced to around 23% to 27%.
- Downsampling plus Merging of depth bits lead to negligible decrease in PSNR of around 0.2dB for natural sequences and 0.2dB for synthetic sequences.
- Downsampling plus Merging of depth bits lead to significant decrease in bitrate of around 17% to 22% for natural sequences and increase of around 1% for synthetic sequences.

- Due to downsampling of depth map, the quality of depth is degraded by around 8dB to 13dB. But, the SSIM results are encouraging as it shows around only 0.01 to 0.02 degradation. Also, visually there is no effect on depth map because the nature depth map is homogenous.
- When 8x8 transform mode is enabled, the amount of compression achieved by texture and depth is 4% to 5% but the merging amount of depth bits is reduced by 22% to 26% compared to merging in only 4x4 transform mode. Thus, enabling 8x8 mode lead to significant decrease in merging.
- View synthesis was carried out on different sequences and it is found out that the degradation in quality of depth map does not affect the synthesized view.

Chapter 6

Conclusion

The robust and scalable coding strategy is developed which ensures that the entire depth map is accommodated in a single bitstream which is fully backward compatible. Simulations show that there is very minimal effect on texture quality (in terms of PSNR) but depth quality is degraded due to upsampling/downsampling of depth map. It has been derived from the subjective evaluation of synthesized views that this degradation of depth PSNR doesn't affect much in the overall rendered quality. So, we strongly believe that the method and the system could be used for FVV/3DV use cases. The approach could be very much effective in terms of encode and/or transmit any metadata including depth maps along with video data. The approach is backward compatible in the sense that any standard decoder can decode the textures for any H.264/MVC use cases (for example, stereo 3D by MVC Stereo High Profile).

The merging concept (using steganography) works better for natural sequences compared to synthetic sequences. Also, the main contribution for the decrease in bitrate is due to downsampling rather than merging. So, we can say that merging technique might not be advantageous in multi view and depth coding. Without applying merging, we can send the downsampled depth appended at the end of every slice and decrease the bitrate of the system with no loss in PSNR. If we consider other applications, this robust coding strategy can be useful to send some other meta data information of the video like camera parameters, etc. As merging amount is considerably decreased while enabling 8x8 mode, it can be derived that this approach might not be applied to HEVC which uses still larger transform blocks. Thus, we can conclude that this proposed approach of steganography based coding can be used as an exploration platform to send some meta data with auto stereoscopic multi view 3D video.

Appendix A

List of Publications

1.) Srijib Narayan Maiti, Parth Desai, Bhargav Patel, Emiliano Mario Piccinelli, Davide Aliprandi, Pasqualina Fragneto, Beatrice Rossi, "Steganography - An approach for video compression", communicated for publication, *IEEE International Conference on Multimedia and Expo(ICME)*, Melbourne, Australia, July 2012.

2.) Srijib Narayan Maiti, Parth Desai, Bhargav Patel, Emiliano Mario Piccinelli, Davide Aliprandi, Pasqualina Fragneto, Beatrice Rossi, "Multiview Video and Depth Coding", to be published, *IEEE Picture Coding Symposium(PCS)*, Krakow, Poland, May 2012.

3.) Srijib Narayan Maiti, Parth Desai, Bhargav Patel, Emiliano Mario Piccinelli, Davide Aliprandi, Pasqualina Fragneto, Beatrice Rossi, "Multiview Video and Depth Coding - Further Research", communicated for publication, *ACM Multimedia*, Nara, Japan, October 2012.

References

[1] Srijib Narayan Maiti, Parth Desai, Bhargav Patel, Emiliano Mario Piccinelli, Davide Aliprandi, Pasqualina Fragneto, Beatrice Rossi, "Steganography - An approach for video compression", communicated for publication, *IEEE International Conference on Multimedia and Expo(ICME)*, Melbourne, Australia, July 2012.

[2] Srijib Narayan Maiti, Parth Desai, Bhargav Patel, Emiliano Mario Piccinelli, Davide Aliprandi, Pasqualina Fragneto, Beatrice Rossi, "Multiview Video and Depth Coding", to be published, *IEEE Picture Coding Symposium(PCS)*, Krakow, Poland, May 2012.

[3] Srijib Narayan Maiti, Parth Desai, Bhargav Patel, Emiliano Mario Piccinelli, Davide Aliprandi, Pasqualina Fragneto, Beatrice Rossi, "Multiview Video and Depth Coding - Further Research", communicated for publication, *ACM Multimedia*, Nara, Japan, October 2012.

[4] Bhargav Patel, "Design and Development of single bitstream containing multiview video and associated depth - Enabling 3D video", M.Tech dissertation, Nirma University, May 2012.

[5] Hakan Urey, Kishore V. Chellappan and Phil Surman, "State of the Art in Stereoscopic and Autostereoscopic Displays", *Proceedings of the IEEE*, Vol. 99, No. 4, April 2011.

[6] Masayuki Tanimoto, Mehrdad P. Tehrani, Toshiaki Fujii and Tomohiro Yendo, "Free-Viewpoint TV - A Review of Ultimate 3DTV and it's related Technologies", *IEEE Signal Processing Magazine*, January 2011.

[7] Thomas Wiegand, Gary J. Sullivan, Gisle Bjontegaard and Ajay Luthra, "Overview of the H.264/AVC Video Coding Standard", *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY*, VOL. 13, NO. 7, July 2003.

[8] Ralf Schfer, Thomas Wiegand and Heiko Schwarz, "The Emerging H.264 Standard", Heinrich Hertz Institute, EBU Technical Review, Berlin, Germany, January 2003.

[9] Iain E. G. Richardson, "H.264 and MPEG-4 Video Compression: Video Coding for Next-generation Multimedia", Wiley Publications, Pg 159-223.

[10] Iain E G Richardson, "H.264/MPEG-4 Part 10: Transform and Quantization", Technical White Paper, March 2003.

[11] P. List, A. Joch, J. Lainema, G. Bjontegaard, and M. Karczewicz. Adaptive deblocking filter, *IEEE Trans. Circuirs Syst. Video Technol.*, VOL. 13, NO. 7, July 2003.

[12] Zhenyu Yang and Yi Cui, "Real-Time 3D Video Compression for Tele-Immersive Environments", Department of Computer Science University of Illinois, Urbana-Champaign.

[13] 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.

[14] Jinwook Choi, Dongbo Min and Kwanghoon Sohn, "2D-Plus-Depth Based Resolution and Frame- Rate Up-conversion Technique for Depth Video", *IEEE Transactions on Consumer Electronics*, Vol. 56, No. 4, November 2010.

[15] Anthony Vetro, Thomas Wiegand, and Gary J. Sullivan, "Overview of the Stereo and Multiview Video Coding Extensions of the H.264/ MPEG-4 AVC Standard", *Proceedings of the IEEE*, Vol. 99, No. 4, April 2011.

[16] Karsten Muller, Philipp Merkle and Thomas Wiegand, "3-D Video Representation Using DepthMaps", *Proceedings of the IEEE*, Vol. 99, No. 4, April 2011.

[17] MPEG2011/N12036, ISO/IEC JTC1/SC29/WG11, "Call for Proposals on 3D Video Coding Technology", March 2011.

[18] Sehoon Yea and Anthony Vetro, "View Synthesis Prediction for Rate-Overhead Reduction IN FTV," IEEE Proc. of 3DTV-CON'08, May 2008.

[19] Shinya Shimizu and Hideaki Kimata, "Improved View Synthesis Prediction Using Decoder-Side Motion Derivation For Multiview Video Coding", *IEEE*, 2010.

[20] Shinya Shimizu, Hideaki Kimata and Y. Ohtani, "Adaptive Filtering in View Synthesis Prediction for Multiview Video Coding", NTT Corporation, *Proceedings of APSIPA*, Japan, October 2009.

[21] P. Merkle, Y. Morvan, A. Smolic, D. Farin, and T. Wiegand, "The Effect of Depth Compression on Multiview Rendering Quality", *IEEE Proc. of 3DTV-CON'08*, May 2008.

[22] Karsten Mller, A. Smolic, K. Dix, Philipp Merkle, and Thomas Wiegand, "Coding And Intermediate View Synthesis Of Multiview Video Plus Depth", *ICIP* 2009.

[23] Philipp Merkle, Aljoscha Smolic, Karsten Mller and Thomas Wiegand, "Multi-View Video Plus Depth Representation And Coding," *ICIP*, 2007.

[24] Manuel Lang, "Nonlinear Disparity Mapping for Stereoscopic 3D", Disney Research Zurich, July 2010.

[25] Shinya Shimizu, Hideaki Kimata and Yoshimitsu Ohtani, "Real-Time Free-Viewpoint Viewer From Multiview Video Plus Depth Representation Coded By H.264/AVC MVC Extension", 3DTV-CON, 2009.

[26] Shinya Shimizu, Hideaki Kimata, Kazuto Kamikura and Yoshiyuki Yashima, "A Backward Compatible 3D Scene Coding Using Residual Prediction", *ICASSP*, 2008.

[27] P. Merkle, Y. Wang, K. Mller, A. Smolic, and T. Wiegand, "Video Plus Depth Compression for Mobile 3D Services", 3DTV-CON, 2009.

[28] D. Stanescu, M. Stratulat, B. Ciubotaro, D. Chiciudean, R. and M. Micea, "Embedding data in video stream using steganography," in *Proc. Int. Symp. Applied Computational Intelligence and Informatics*, pp. 241-244, May 2007.

[29] Y. Hu, C. Zhang and Y. Su, "Information hiding based on intra prediction modes for H.264/AVC," in *Proc. IEEE Int. Conf. Multimedia and Expo*, pp. 1231-1234, Jul. 2007

[30] C. Xu, X. Ping and T. Zhang, "Steganography in compressed video stream," in *Proc. IEEE Int. Conf. Innovative Computing, Information and Control*, Beijing, vol. 1, pp. 269-272, Aug. 2006.

[31] Mansouri Jafar and Khademi Morteza, " An adaptive method for compressed video steganography using spatial and temporal features of the video signal", in *Proc ICEE*, Iran, Vol. 4 , May 2009.

[32] A.M. Bruckstein, M. Elad, Ron Kimmel, "Down-scaling for better transform compression," *IEEE Transactions on Image Processing*, vol.12, no.9, pp. 1132-1144, Sept. 2003.

[33] Erhan Ekmekcioglu, Stewart T. Worrall, Ahmet M. Kondoz, "Bitrate adaptive downsampling for the coding of multiview video with depth information", *3DTV-CON'08*, May 2008.

[34] Qiuwen Zhang, Ping An1, Yan Zhang1, Qian Zhang1, Zhaoyang Zhang1, "Reduced Resolution Depth Compression for Multiview Video plus Depth Coding", *ICSP2010 Proceedings*, July 2007.

[35] ITU-T Recommendation H.264 standard, Advanced video coding for generic audiovisual Services, March 2003.

[36] MVC Software Manual for JMVC, version 7.0, Dec 2009. Available mvclab.googlecode.com/ files/SoftwareManual.doc.

[37] Zhou Wang, Alan Conrad Bovik, Hamid Rahim Sheikh, Eero P. Simoncelli, "Image Quality Assessment: From Error Visibility to Structural Similarity", *IEEE Transactions on Image Processing*, Vol. 13, No. 4, April 2004.

[38] Davide Aliprandi, FVVSynthesis Software, version 4.0, Available STMicroelectronics Repository.

Index