End-to-End 3D Video Communication over Heterogeneous Networks

A thesis submitted for the degree of

Doctor of Philosophy

by

Hamdullah Mohib

Supervised by Prof. Abdul H. Sadka

Electronic and Computer Engineering

School of Engineering and Design

Brunel University London

April 2014

Abstract

Three-dimensional technology, more commonly referred to as 3D technology, has revolutionised many fields including entertainment, medicine, and communications to name a few. In addition to 3D films, games, and sports channels, 3D perception has made tele-medicine a reality. By the year 2015, 30% of the all HD panels at home will be 3D enabled, predicted by consumer electronics manufacturers. Stereoscopic cameras, a comparatively mature technology compared to other 3D systems, are now being used by ordinary citizens to produce 3D content and share at a click of a button just like they do with the 2D counterparts via sites like YouTube. But technical challenges still exist, including with autostereoscopic multiview displays. 3D content requires many complex considerations--including how to represent it, and deciphering what is the best compression format--when considering transmission or storage, because of its increased amount of data. Any decision must be taken in the light of the available bandwidth or storage capacity, quality and user expectations. Free viewpoint navigation also remains partly unsolved. The most pressing issue getting in the way of widespread uptake of consumer 3D systems is the ability to deliver 3D content to heterogeneous consumer displays over the heterogeneous networks. Optimising 3D video communication solutions must consider the entire pipeline, starting with optimisation at the video source to the end display and transmission optimisation. Multi-view offers the most compelling solution for 3D videos with motion parallax and freedom from wearing headgear for 3D video perception. Optimising multi-view video for delivery and display could increase the demand for true 3D in the consumer market. This thesis focuses on an end-to-end quality optimisation in 3D video communication/transmission, offering solutions for optimisation at the compression, transmission, and decoder levels.

Acknowledgements

This thesis would not have been possible without the support of my my family, friends and colleagues. I would like to thank them for their unconditional support and encouragement throughout this sometimes difficult process.

Special thanks to my supervisor for his personal and professional guidance throughout my program. I also owe my wife a special debt of gratitude for her love, support and patience and for being there when I thought of giving up. I also want to thank my new daughter whose smiles were the perfect remedy for a tough day.

I would also like to thank Brunel University for funding my research through the Isambard Research Scholarship. I am also thankful to the research office staff for their support and guidance every step of the way.

List of Abbreviations

2D	Two-dimensional (video/image)
2DTV	Two-dimensional Television
3D	Three-dimensional (video/image)
3DTV	Three-dimensional Television
4kHD	4K Resolution High Definition (video)
AS3D	Autostereoscopic three-dimensional (video)
AVC	Advanced Video Coding
CNAME	Canonical Names
DBIR	Depth-based Image Rendering
DCCP	Datagram Congestion Control Protocol
DSL	Digital Subscriber Line
DVB-H	Digital Video Broadcasting - Handheld
DVB-T	Digital Video Broadcasting - Terrestrial
DVD	Digital Video Disc
FP6	(EU) Framework Program (for research and Technological development) 6
FP7	Framework Program 7
FTV	Free-viewpoint Televisions
GOP	Group of Pictures
GOV	Group of Views
H3D	Holoscopic 3D
HD	High Definition
HDMI	High Definition Media Interface
HTTP	Hypertext Transport Protocol
HVS	Human Visual System
IDR	Instantaneous Decoding Refresh
IP	Internet Protocol

IPTV	Internet Protocol Television
ISO/IEC	International Organisation for Standardisation/International Electrotechnical Commission
ITU-T	International Telecommunication Union – Telecommunication Standardisation sector
JPEG	Joint Photographic Experts Group
JVT	Joint Video Team
LED TV	Light Emitting Diode Television
M3D	Multi-view 3D
Mbps	Megabits Per Second
MERL	Mitsubishi Electric Research Laboratories
MOS	Mean Opinion Score
MPEG	Moving Picture Experts Group
MPEG-2 TS	MPEG-2 Transport Stream
MST	Multi-Session Transmission
MTU	Maximum Transmission Unit
MVC	Multi-View Video Coding
NAL	Network Abstraction Layer
NALU	NAL Unit
PC	Personal Computer
PLPMTUD	Packetisation Layer Path MTU Discovery
PPS	Pictures Parameter Set
PSNR	Peak Signal-to-Noise Ratio
QoE	Quality of Experience
QP	Quantisation Parameter
RANSAC	Random Sample Consensus
RFC	Request for Comments

RTP	Real-time Transport Protocol
SEI	Supplemental Enhancement Information
SIFT	Scale-invariant Feature Transform
SITL	System-in-the-loop
SPS	Sequence Parameter Set
SSD	Sum of Squared Differences
SST	Single Session Transmission
SVC	Scalable Video Coding
TCP	Transport Control Protocol
T-DMB	Terrestrial-Digital Multimedia Broadcasting
UDP	User Datagram Protocol
Ultra-D	Ultra-D is a proprietary 3D technology
Ultra-HD	Ultra High Definition (television)
UMTS	Universal Mobile Telecommunications System
VCL	Video Coding Layer
V-IDR	View-Instantaneous Decoding Refresh
VQEG	Video Quality Experts Group
VSP	View Synthesis Prediction

Table of Contents

Table of Figures

Chapter 1: Introduction

This chapter introduces the content and layout of the thesis, and the aims and objectives of the research. In addition, it provides a list of the research publications made by the author and summarises the research contributions made in each chapter. A summary of the content of the following chapters is also presented.

1.1 Overview

3D videos have gained considerable attention from the general public in the last decade. Major blockbusters are grossing an ever increasing amount of their revenue from 3D films. In the past few years, 3D displays have made it to people's living rooms. There are exclusive 3D channels broadcast. 3D capturing cameras are now widely available for amateur film makers and hobbyists to create their own 3D videos. Video hosting sites such as YouTube now allow the general public to upload and share their videos in 3D.

However, there is still a long way to go. First of all, the 3D videos created are mostly stereoscopic, which requires users to wear a headgear to perceive 3D. In addition to the discomfort of wearing these headgears, stereoscopic videos cause eye-fatigue when watched for too long. There is still a vacuum of content available to the end users. Transmission of 3D video content requires more bandwidth compared to its 2D predecessor because of the additional view(s).

The lack of motion parallax in stereoscopic 3D videos is another disadvantage that prevents the viewer from fully immersing in the video. Multi-view auto-stereoscopic 3D videos solve the parallax problem but increase the amount of video content that is

proportionate to the number of additional views. In addition, the capture of multi-view videos requires complex camera calibration and synchronisation.

While the broadcast of 3D videos over existing terrestrial networks is already in place and practised in many developed countries, the video quality is compromised in order to fit the stereoscopic content into the existing channels, which were designed for high definition 2D videos. There are still gaps in designing a practical transmission scheme for multi-view video content.

Internet, a heterogeneous network of networks, with increasing bandwidth and speeds, has already changed people's viewing habits. People like to watch what they want and when they want to instead of waiting for strict broadcast schedules. The concept is commonly referred to as On-Demand TV. Most broadcasters now have their own On-Demand streaming options available online, such as iPlayer, 4oD, and Demand 5 from BBC, Channel 4, and Channel 5, respectively, to name a few. There are independent On-Demand video content providers that are now competing with broadcasters such as NetFlix, LoveFilm, BlinkBox and Now TV. This change is reflected in the new Smart TVs which come with internet connectivity for on demand content.

Moreover, other immersive displays are emerging in the market, including the 4kHD and Ultra-HD. These displays provide at least 4 times the current high definition video quality. The production, transmission, and display of such systems is far less complicated than 3D videos.

For 3D videos to be able to compete with such systems and to secure its future, it must be able to offer a competitive advantage. While 3D perception might be the unique

selling point, stereoscopic videos will not deliver that market share due to the restrictions and reduced quality of videos associated with it. 3D videos must provide glasses-free and comfortable viewing with an immersive motion parallax. Multi-view video with auto-stereoscopic displays could be one potential solution, if streaming was easy and straightforward, and content generation was not so complicated and expensive. The added benefit of allowing users to change their viewing angles in free-view-point TV (FTV) adds to its competitive advantage.

This research focuses on end-to-end quality optimisation in 3D videos over heterogeneous networks. This research analyses the different 3D video systems currently available and under research. It explores the most promising technology—multi-view 3D videos—from compression, transmission and display angles. The complete pipeline, shown in *Figure 1.1*, is considered to find solutions for 3D video optimisation and delivery, from choosing the correct codec and transmission methods to reducing the number of cameras needed for capturing 3D video content.

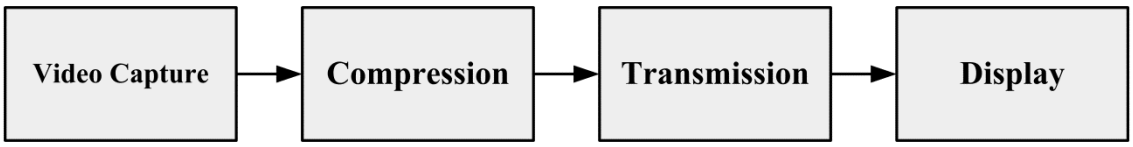


Figure 1.1: 3D video communication pipeline.

In the **video capture** stage, the most pressing challenge is the complexity of camera setup which is directly proportionate to the number of cameras needed. The challenge at the *compression* stage is related to the challenge of the video capture. We have a large amount of data that needs to encoded. The *solution* presented in this thesis for

these two stages is the utilisation of virtual view rendering techniques to reduce the number of cameras. This directly affects the compression stage, increasing the efficiency of the *multi-view video coding*, which as demonstrated in chapter 3 efficiently exploits inter-view redundancies to reduce the amount of data to be stored/transmitted with negligible effect on video quality.

Transmission errors in one view of a multi-view video propagate to other views which negatively affect the video quality. One of the *solutions* presented in the thesis is the use of multi-session transmission to contain errors only within parts of the bitstream that can easily be recovered or to frames that do a negligible effect on the video quality. This solution is complemented by the study of variable packet sizes in congested wireless networks which is also presented in this thesis.

3D **displays** currently available include the multi-view autostereoscopic systems that do not require the user to wear glasses for 3D perception. It has the potential for widespread adaptation by the general public. However, due to the lack of content for such displays, their use is very limited. The *solutions* presented in this thesis are aimed at the increased adaptation of the multi-view autostereoscopic displays by means of reducing the complexity of video capture, increasing the coding efficiency and improving the error resilience of the multi-view video bitstream during transmission.

1.2 Research Aims

The aim of this research is to analyse the 3D video transmission pipeline and optimise each section from end-to-end. The process starts with the compression of 3D videos and the current state-of-the art codec. The choices made at this stage affect the whole

pipeline, including transmission and display. The transmission over heterogeneous networks is then analysed, and different protocols are tested, before moving to codec optimisation. The end-to-end analysis of the transmission pipeline gives us the full picture and puts us in a unique position to offer optimisation solutions for the entire pipeline. Only by understanding the full transmission pipeline can one understand the specific challenges and study the opportunities available.

Multi-view video offers numerous advantages and is, currently, the only viable option for implementing free-view-point TV. Optimising the compression, transmission and display of multi-view videos will pave the way for increasing the demand for 3D videos and improve the quality of viewing experience.

The following list summarises the aims and objectives of the research:

- Survey the state of the end-to-end video systems and how they work together to provide users quality of experience;
- Understand the challenges in 3D video communication and devise solutions to resolve them;
- Analyse compression techniques currently available and understand how they affect the communication pipeline in 3D videos;
- Study the transmission of 3D videos over heterogeneous networks and offer solutions that could improve the transmission eco-system;
- Devise solutions for the entire pipeline taking the encoder/decoder, transmission and displays into account.

1.3 Publications

The publications published during this research are listed below.

1.3.1 Journal Publications

I. Nawaz, Muhammad, John Cosmas, Pavlos I. Lazaridis, Zaharias D Zaharis, Yue Zhang, and **Hamdullah Mohib**, "Precise Foreground Detection Algorithm Using Motion Estimation, Minima and Maxima Inside the Foreground Object," *Broadcasting, IEEE Transactions on* , vol.59, no.4, pp.725,731, Dec. 2013

II. **Mohib, Hamdullah**, Abdul H. Sadka, "Multi-Session Transmission of Multi-view Video", Electronic Letter, IET, (submitted)

III. **Mohib, Hamdullah**, Abdul H. Sadka, "Decoder based virtual view rendering for multi-view video", Image and Vision Computing, Elsevier (submitted)

1.3.2 Conference Papers

I. **Mohib, Hamdullah**, Abdul H. Sadka, and Mohammad R. Swash. "Multi-session transmission of H. 264/MVC over heterogeneous IP networks." In *Broadband Multimedia Systems and Broadcasting (BMSB), 2013 IEEE International Symposium on*, pp. 1-6. IEEE, 2013.

II. **Mohib, Hamdullah**, Mohammad R. Swash, and Abdul H. Sadka. "Multi-view video delivery over wireless networks using HTTP." In *Communications, Signal Processing, and their Applications (ICCSPA), 2013 1st International Conference on*, pp. 1-5. IEEE, 2013.

III. Sadka, Abdul, **Hamdullah Mohib**, Abdulkareem Ibrahim, and Mohd. M. Salzali, "Compression of 3D Stereoscopic Video Using ITU-T H.264/AVC" In *Second Abu Dhabi University Annual Research Conference*, 2013

IV. **Mohib, Hamdullah**, and Abdul H. Sadka, "Quality Optimisation in 3D Video Communication over Heterogeneous Networks", in *Research Student Conference School of Engineering and Design, ResCon12, Brunel University*, 2012

V. **Mohib, Hamdullah**, and Abdul H. Sadka, "The Effect of Transmission Errors in IP Networks On the Perceived Quality of Multi-view Video", *Research Student Conference School of Engineering and Design, ResCon13, Brunel University*, June 2013

1.4 Research Contributions

The research contributions made in this thesis are listed below, in the order of chapters in this thesis:

I. A detailed literature survey of 3D video systems with current challenges.

II. Comparison and evaluation of simulcast compression and multi-view compression techniques.

III. Investigation of the effects of variable packet sizes on transmission in congested wireless IP networks.

IV. Investigation of multi-session transmission in multi-view video for increased protection against transmission errors and adaptive video transmission depending on channel conditions and end-point decoding capabilities.

V. Utilisation of virtual view rendering techniques to improve multi-view codec efficiency and evaluation of their execution complexities.

1.5 Thesis Outline

This thesis is organised into seven chapters and five appendices. Chapter 1 lays the foundation for the entire thesis. It introduces the research topic, its aims and objectives. It also provides an overview of the research contributions and lists the research publications and conferences attended.

Chapter 2 provides an overview of end-to-end 3D video systems. Through an extensive literature survey, it lists the contribution of the research community in the area of 3D videos and the communication pipeline. The chapter starts with providing an understanding of how 3D is perceived and the different techniques used to represent 3D videos. It then explains the state-of-the-art in 3D displays, including the displays that are currently under research. It then discusses the transmission of 3D videos and the contributions the research community has made to this area. A detailed survey of 3D video transmission in terrestrial broadcast is given, followed by an introduction to transmission over the internet. Challenges that the 3D video systems currently face are discussed.

Chapter 3 provides an understanding of compression techniques in 3D videos. The current state-of-the-art codec, the Multi-view Video Coding (MVC) extension of the H.264 Advanced Video Coding (AVC), is explained in relation to coding efficiency and transmission techniques. MVC is highly adaptable and provides a very efficient

coding functionality. The contribution in this chapter is the comparative evaluation of different compression techniques for coding efficiency.

Chapter 4 discusses 3D video streaming over the internet using the HTTP protocol using simulated networks. 3D video streaming is introduced and the advantages of using the Hypertext Transfer Control Protocol (HTTP) are discussed. A variable packet size application was developed to transmit multi-view videos over HTTP using packet size variation. OPNET's System-in-the-loop (SITL) module is used for the communication to real end points over a simulated heterogeneous network. Various packet sizes are tested under interface free and congested network conditions. The contribution made in this chapter is the experimental investigation of the effects of variable packet sizes on transmission in congested wireless IP networks. The segmentation at the application layer has a direct effect on transmission times. It was found that that 64 kB is the optimum segment size. An application was developed to enable the selection of packet size when the client requests the video from the server over the simulated network.

Chapter 5 further investigates the transmission of multi-view videos over Real-time Transmission Protocol (RTP) for real time video communication. The key contribution of this chapter is the study of the effect of transmission errors on the multi-view video codec is and a solution using multi-session transmission (MST) is proposed to decrease the effect of errors on the coded bitstream. Modifications to the MVC decoding frame management is made to increase codec robustness. Sirannon, a testing tool accepted by the Video Quality Expert Group (VQEG), is used for the multi-session transmission and error injection into the bitstream. Both single session transmission and multi-

session transmission are tested for ten packet loss scenarios using ten transmissions each, in order to increase the accuracy of the test results. The transmission of the H.264 multi-view video coding (MVC) bitstreams over multiple streams based on priority and view changes improves the quality of 3D video. MST of multi-view video can render the video stream more adaptable to channel conditions and offer additional error resilience capabilities by confining the errors to a particular spatial region. Experimental results prove MST's effectiveness in handling error rates as high as 10% with negligible frame drop when the errors in transmission are confined to the relatively less important bi-predicted frames.

Chapter 6 investigates MVC codec optimisation by applying virtual view rendering of views at the decoder, using two adjacent decoded views. Hence, increasing the coding efficiency of the video codec and decreasing the need for the number of cameras needed to capture multi-view videos. The key contribution in this chapter is the evaluation of three techniques for virtual view rendering and their execution complexity, which is an important feature in real time video playback. The three methods, which use simple image stitching, disparity prediction and panoramic view creation techniques are analysed and the results compared through objective and subjective testing. The challenges with virtual view rendering are also discussed in detail, with examples.

Chapter 7 concludes the investigations of this research and makes recommendations for future work.

Results beyond the ones present in this thesis, program codes and software manuals are in **Appendices A to D**.

1.6 Conclusions

This chapter summarises the research. It starts by explaining the problem statement of the research, and continues to highlight the aims and objectives of the thesis. It is followed by a list of research publications and a list of research contributions made in this thesis. The layout of the thesis is explained with a brief description of the content of each chapter.

The next chapter presents a detailed literature survey of the 3D eco-system and the challenges that are currently unresolved.

Chapter 2: 3D Systems and the Communication Pipeline

Three-dimensional technology, more commonly referred to as 3D technology, has revolutionised many fields, including entertainment, medicine, and communications to name a few. In addition to 3D films, games, and sports channels, 3D perception has made tele-medicine a reality.

Thirty percent of the all HD panels at home by the year 2015 will be 3D enabled, as predicted by consumer electronics manufacturers. Stereoscopic cameras, a comparatively mature technology compared to other 3D systems, are now being used by ordinary citizens to produce 3D content and share at a click of a button just like they do with 2D counterparts via sites like YouTube.

But technical challenges still exist, such as with autostereoscopic multiview and holoscopic video displays. Free viewpoint navigation also remains partly unsolved [1]. The most pressing issue getting in the way of widespread uptake of consumer 3D systems is the ability to deliver 3D content to heterogeneous consumer displays over the internet.

This chapter sets the scene by introducing 3D, showing how it is represented, stored and transmitted, and the different types of displays that can be used to perceive 3D perception, as well as the challenges that need to be addressed.

2.1 Introduction

The concept of 3D is not new. As early as 300 BC, Euclid discovered that humans see slightly different images of the same object with each eye. Eleventh century mathematician Al-Hazen also explained the concept of 3D in his book of optics. Since then, there has been gradual development with 3D research over the years until today. Today, 3D films gross more than ever before in cinemas, 3D sports channels are on satellite TV and video streaming online is available.

The evolution of video processing technology is driven by progress in technological advancements and the content generation capabilities to meet user demands. The evolutionary process began with analog TV and is currently at the smart TV stage, which combines HDTV, IPTV and in some displays 3DTV [2]. 4K HDTV is the next stage in the flat screen TV evolution, currently an active area of research [3]. In 3DTV, the most prominent technology in use is the stereoscopic 3D displays.

The video processing field has evolved from one video displayed at different resolutions. The needs of the current era require on-demand connected interactive videos where the social experience is at the heart of it. The need for 3D video processing has given rise to content-adaptive processing that includes segmentation, depth map extraction, and structure from motion. [2]

Smart TV is indicative of the change in people's viewing habits. TV has exceeded the processing expectations from a PC in the 1990s. The Smart TV does not only support improved performance in terms of display, but is more efficient in power consumption, has more computing power, comes with a large amount of memory, has built-in

interactivity, and has internet connectivity at its heart [2]. In the race towards a true free view point TV, in many ways, the Smart TV in itself predicts the future of what will be expected from multi-media transmissions. It is no longer a straight forward broadcast or even multi-cast. Viewers already expect on demand content in 2D, thus their expectations from 3D videos will be threefold: on demand, interactive view angle changes, and high-definition quality, mobility, and seamless 2D to 3D conversion. Thus, it is imperative that any 3D communication system must meet these expectations.

This chapter discusses the visual perceptions created to generate 3D, the displays currently available and under research, and finally how they are delivered and the challenges in the 3D communication pipeline.

2.2 Fundamentals of 3D

3D perception requires eyes and the brain, together referred to as the human visual system (HVS). They work together to receive light rays and interpret them to create the image seen in three dimensions [4]. Perception, it must be clarified, is not seeing the real world in our daily life. The HVS is fed the data that would make it look like the real world for the most part, but it fails to deliver the required outcome if the person perceiving 3D does not fit the conditions such as having irregular or defected eyes. Humans perceive 3D in two different ways; one is through binocular parallax, where we see a different angle of the same object through each eye. The other is motion parallax, where we see a different angle of the same object by moving our head or the movement of the object [5]. Both techniques were discovered as early as the 19th century by Wheatstone in 1838 and Helmholtz in 1866, respectively [6]. Wheatstone

demonstrated stereopsis, the unique depth sense produced by retinal disparity through constructing a simple stereoscope from mirrors.

Stereoscopic 3D perception has come a long way since Wheatstone first proved the link between parallax and depth perception by studying them using a stereoscope, considered to be the world's first 3D display [6]. Stereoscopic perception exploits binocular disparity in the human visual system. Two slightly different images are presented, one to each eye, captured by identical cameras that are placed 65 millimetres apart to correlate with the average distance between eyes [7]. The two images, fused in the visual cortex of the brain, compose a three-dimensional view. The techniques used to feed different images to the eyes range from using headgear (glasses) to using parallax barriers in the screen.

2.2.1 3D Perception through Glasses

The three most commonly used glass technique are anaglyph, polarisation and time division. Anaglyph glasses use colour coded image projection to separate the views that are simultaneously fed to the left and the right eye, to create the depth perception [8]. The second method, most commonly used today, is polarised glasses which simultaneously filter the left and the right views through linear or circular polarisation [9]. Anaglyph and polarisation techniques are shown in Figure 2.1.

While both the anaglyph and polarisation techniques filter different light rays simultaneously, the time division technique, more commonly referred to as active shutters, uses frame sequencing to feed the left and the right images to the eyes one at a time, as depicted in Figure 2.2. While one eye sees an image the other eye does not see

anything. Active shutter glasses need a higher frame rate compared to anaglyph and polarisation techniques and require the glasses to synchronise with the screen [11].

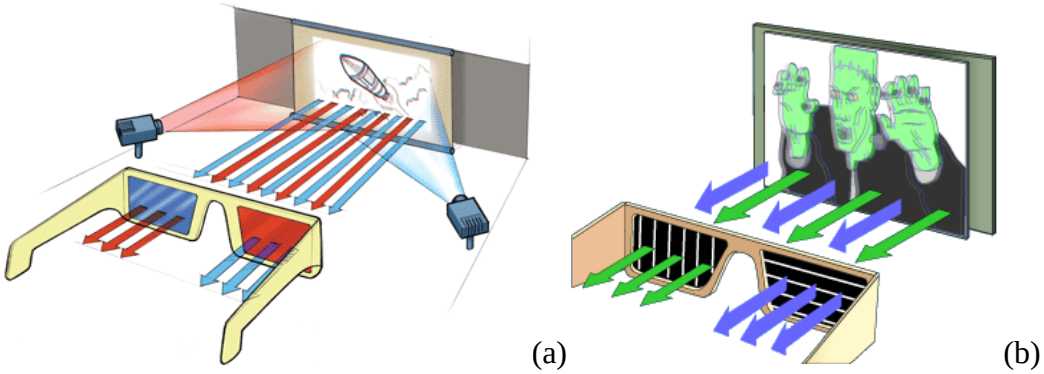


Figure 2.1: 3D perception using (a) anaglyph Filtering and (b) Polarised Filtering Techniques [10]

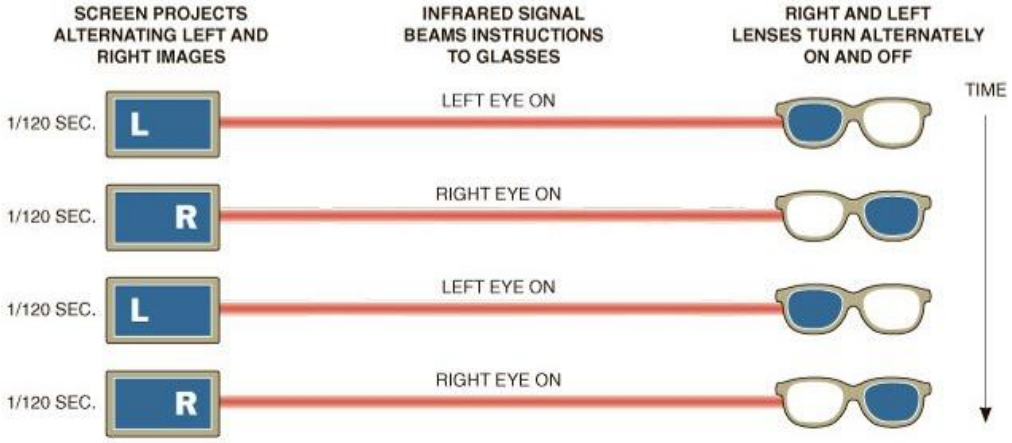


Figure 2.2: Demonstration of time-division technique (active-shutters) for displaying 3D videos [12]

2.2.2 Glasses Free 3D perception

Perceiving 3D with the help of glasses, regardless of the technique, presents several drawbacks. In addition to having to wear glasses, the most common drawbacks include

eye-fatigue, cross-talk of various degrees, and the lack of any motion parallax. 3D systems that do not require glasses utilise lenticular sheets or parallax barriers to direct the light rays to the left and right eyes, and are commonly called auto-stereoscopic as shown in Figure 2.3.

Multi-view auto-stereoscopic systems, which provide a binocular vision from any angle without any special glasses, were first introduced by Stephen Benton in the 1970s—who refined the vision of 3D displays by Gabriel Lippmann in 1908 [6]. Multi-view replicates stereoscopic systems but offer more views to add motion parallax [14]. Viewers will still see two views at a time but by moving the head, the viewer sees a different angle, hence providing a more realistic 3D experience.

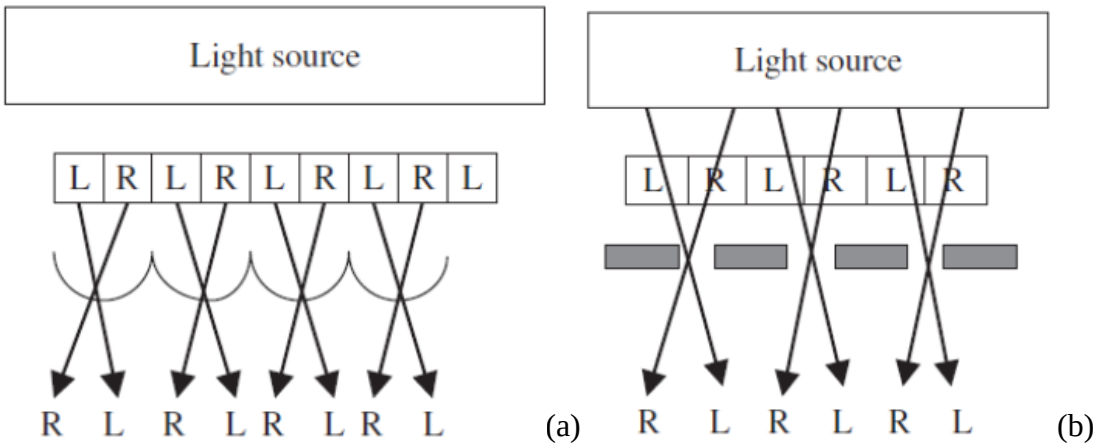


Figure 2.3: Redirecting the light to the correct eye in the auto-stereoscopic technique: (a) lenticular sheet and (b) parallax barrier [13]

2.3 3D Video Representation

There are two key issues with 3D content: how to represent it and what is the best compression format, when considering transmission or storage, because of its increased

amount of data. Any decision must be taken in the light of the available bandwidth or storage capacity, quality and user expectations. There are typically three representation formats namely: full resolution, frame compatible resolution, and depth based representation [15].

2.3.1 Full Resolution Representation

This takes the full HD resolution videos, typically 1920x1080 or 1280 x 720, for all views starting with two views for stereo to N views for multi-views [15]. This means that the raw data generated is directly proportionate to the number of views captured, as shown in Figure 2.4. Stereo cameras that capture the left and right views at half the HD resolution are not considered full resolution.

The full resolution format produces the best quality results because none of the views are degraded to save bandwidth or storage capacity.

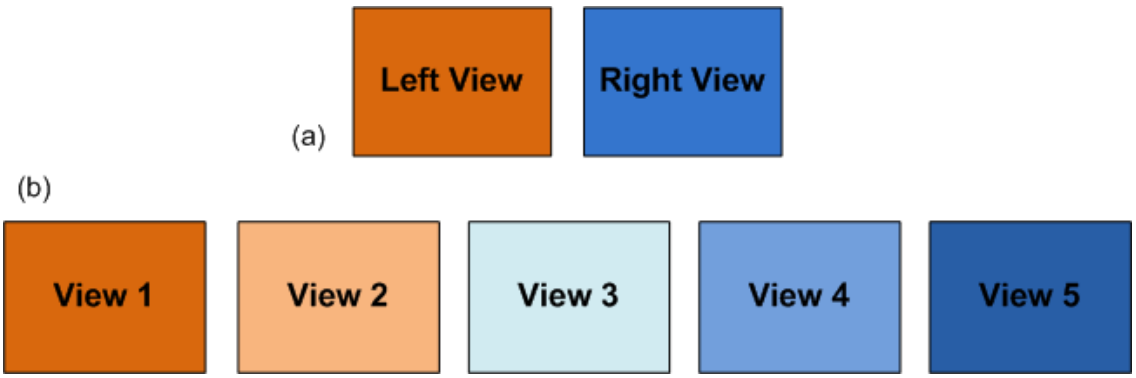


Figure 2.4: In the Full Resolution Representation all views are taken as they are: (a) both left and right views in Stereoscopic 3D representation are taken as full, (B) In multi-view all views are taken, in this example there are 5 views. They are all represented

2.3.2 Frame-Compatible Representation

In stereo 3D videos, the left and the right views are sub-sampled and interleaved in to a single frame or sequence of frames [15]. The multiplexing reduces the amount of data to take advantage of existing infrastructure for broadcast. The sub-sampling can be done either (i) horizontally and placed top-and-bottom—proposed by Comcast; or (ii) vertically and placed side-by-side—proposed by Sansio, RealD and adopted by Samsung, Panasonic, Sony, Toshiba, and Direct TV [13]; or (iii) checkerboard format; or (iv) temporal multiplexing—where the frames or fields are placed sequentially after one another at half the frame rate for each view, to equate the data of a single view; or (v) as mixed resolution—where one view has a reduced resolution. Depictions of the frame-compatible representations are shown in Figure 2.5.

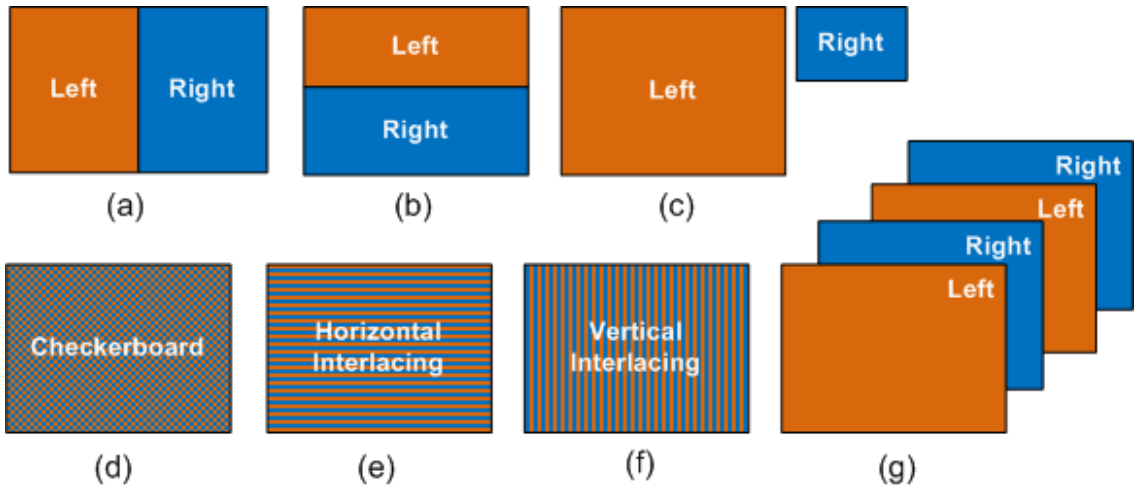


Figure 2.5: Frame compatible representation of 3D stereoscopic videos: (a) Left-Right Representation, (b) Top-Bottom, (c) Mixed Resolution, (D) Checkerboard Interlacing, (e) Horizontal Interlacing, (F) Vertical Interlacing, (g) Temporal (Time Division) representation

Frame-compatible representation makes use of existing encoder/decoders and transmission channels making it an ideal solution for quick deployment over existing infrastructure at the cost of reduced quality in spacial or temporal resolution.

2.3.3 Depth-based Representation

Depth-based-image-rendering (DBIR), technique helps in generating a virtual view, effectively reducing the amount of data for transmission or storage, typically called 2D plus depth [15]. Depth can be captured by the camera or can be extracted from a stereo pair by solving for stereo correspondence stored in a format called the depth map. The depth map is used by the decoder to extract the second view. While it can be used for both stereo and multiview videos, 2D plus depth format cannot handle occlusion very well and is limited in the amount of depth range it can render [16]. Using depth data, instead of the full view, has its benefits of reducing data for storage or transmission. On average, it requires only 10 to 20% of the original view for storage or transmission [17].

Figure 2.6: Example of 2D plus depth format [17]

2.4 3D Compression Formats

The 3D representation formats presented above can be compressed using different methods. Sometimes the representation format reduces the choices of compression but in most cases, the following compression techniques apply to all. They are briefly introduced here and discussed in the future chapters in more details.

2.4.1 Simulcast

In simulcast, each view is encoded independently, without taking interview similarities in to account [15]. It helps keep the computational requirements low during encoding/decoding and is inherently backwards compatible. However, it means the amount of data to be stored and transmitted will increase proportionally depending on the number of views. The only way to reduce the data is to either encode one of the views with a less quality or smaller spacial resolution as described above in the frame compatible representation. The key difference in simulcast compression format is that each view is encoded independently and hence can be transmitted without any effect on the other view(s). Simulcast is discussed in Chapter 3 in more details.

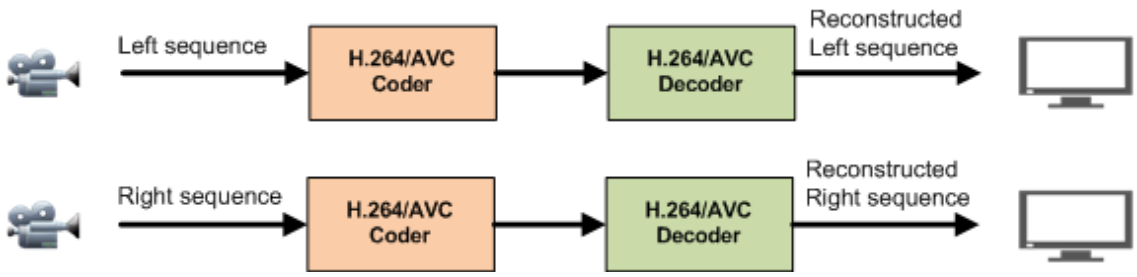


Figure 2.7: Example of simulcast coding using H.264/Advanced Video Coding (avc)

2.4.2 Frame Compatible Coding with SEI Message:

Supplemental Enhancement Information (SEI) message, in H.264/AVC standard, are transmitted with frame-compatible videos to signal the packaging arrangements between them [15]. It helps the decoder identify the order of the images, scaling required, colour format conversion, and de-noising if required. Receiving devices can be signalled through SEI message of the frame-compatible format through supported interfaces such as the High-Definition Multimedia Interface (HDMI) [15]. The videos are encoded as one video. This compression technique benefits from a reduced amount of data and, because the videos are merged before compression process, redundancies are accounted for, hence producing an efficiently compressed bitstream. However, the quality degradation is increased because each view is taken at half the resolution as shown in Figure 2.8.

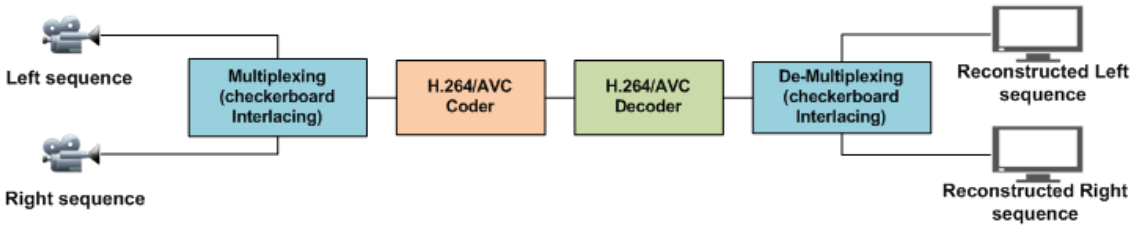


Figure 2.8: Example of frame compatible coding with SEI messages using checkerboard representation and H.264/AVC encoder

2.4.3 Multi-View Video Coding (MVC)

A key concept of video compression is the exploitation of redundancies between consecutive frames. In 3D videos the redundancies between views can also be exploited to increase encoding efficiency. The MVC extension of the H.264/Advanced Video Coding (AVC) exploits these redundancies, reduces decoding complexity, and adds

new functionalities specific to multi-view video coding [18]. MVC can be used to encode full resolution formats, mixed resolution, and video plus depth depending on the requirement. Figure 2.9 shows an example of multi-view video coding for three views. MVC is discussed in more details in Chapter 3.

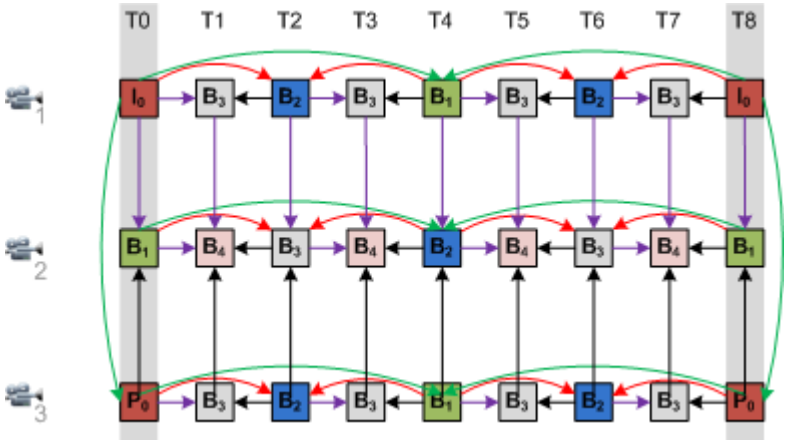


Figure 2.9: Example of H.264/MVC compression technique for three views

2.5 3D Displays

3D display manufacturers are continuously trying to produce realistic 3D displays. Most of the displays currently in production are stereoscopic TVs with active shutter glasses. Our extensive search of 3D display markets show that the most popular products on the UK market are Sony Bravia Smart 3D LED TV, Samsung 3D Plasma TV, LG Smart 3D LED TV and Panasonic Viera Smart 3D LED TV. They all come with a various number of free glasses. [19][20] Sony and LG come with active shutter glasses where as the Samsung and Panasonic with polarised glasses. Auto-stereoscopic displays are still rare and a novelty product but certainly available to buy if one wishes. Toshiba, Sony, Aliscopy and Philips have auto-stereoscopic TVs available for sale,

albeit at a high cost. Some manufacturers, such as Toshiba's Qosimo F755 [21], have also employed auto-stereoscopic abilities in laptop displays.

Some manufacturers such as Tridelity have multi-view displays capable of displaying five views. [22] This section describes the different techniques used to display 3D videos. Some of them such as volumetric displays are still in experimental stages but it is important to understand what the future holds for 3D displays.

2.5.1 Stereoscopic displays

Stereoscopic displays employ one of the 3D representation techniques described above. They require the viewers to wear special glasses that filter the left and the right view to perceive depth. They each employ a different technique to do this, either by overlaying the images or showing them alternatively in fast succession. These glasses, namely, anaglyph, polarized or active-shutters glasses [2], ensure the viewers only see the left eye image with the left eye and the right eye image with the right eye.

These displays, also referred to as binocular displays, use a particular multiplexing method to display videos. The multiplexing combined with the glasses, which could either be active or passive, enables the user to perceive 3D depth [23].

Anaglyph display method uses colour multiplexing method to combine stereo pairs overlaid onto each other by using non-overlapping colours. The most common colour pairs used are red and blue, red and green, red and cyan, and amber blue and dark blue. Anaglyph displays, due to its use of colours to separate left and right views, affect the natural colour of the image. It also suffers from cross-talk, which in the specific case of

Anaglyph means, the left and right views leaking over to each other. This can result in eye-fatigue [24].

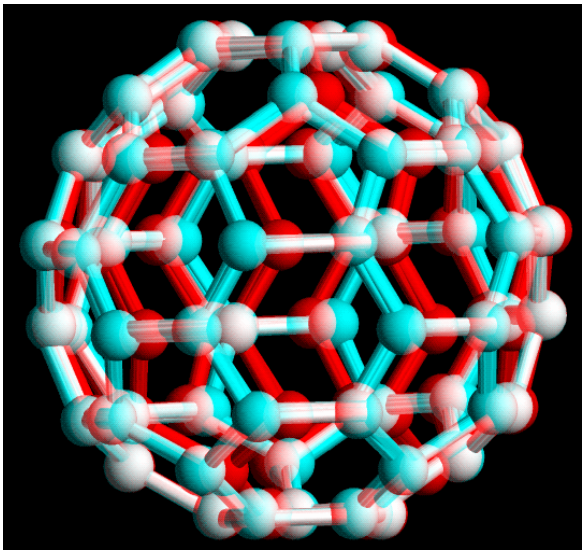


Figure 2.10: An example of combining left and right views in anaglyph format for display [25]

Figure 2.11: A range of different anaglyph glasses [26]

Polarized display method uses polarization-division multiplexing, where, just like anaglyph, two non-overlapping polarised images are laid on top of each other. This polarisation can be, as described above, linear or circular. It is filtered by matching polarized filtered glasses. Linear polarisation, horizontal or vertical, suffers from image distortion when the viewers tilt their head. Circular polarisation solves this problem to give the user seamless 3D perception from any angle [27].

Figure 2.12: An example of polarised glasses [28]

Active-shutter display systems use time-division multiplexing to display the right and left images to the viewer consecutively one after another. This method requires the glasses to work in sync with the display, hence the name active-shutter, to display one image to one eye while masking the other. This method requires working at least twice the frame rate of normal video to make up for the alternation [11].

Figure 2.13: A range of Active-shutter glasses provided by Samsung with their 3D Displays [28]

Figure 2.14: Example of a head mounted display use [29]

2.5.2 Head Mounted 3D Displays

Head mounted displays, such as the Sony Personal 3D viewer [29], are similar to stereoscopic displays. They require the user to wear a headgear that directs the left and the right views to each eye in a more confined space.

2.5.3 Volumetric 3D Displays

Volumetric 3D displays have better 3D perception than stereoscopic techniques. They use projected light emission in a confined volume to generate a 3D object in space [30].

Volumetric displays can be generated using different techniques [31], from cylindrical displays [32], such as the one shown in Figure 2.15, to stacking multiple 2D displays that each displays a slice of the scene to create 3D effect [24].

Volumetric displays offer the real world experience. The objects inside them can be seen from 360 degree angles simultaneously by as many viewers as can fit around the display with full parallax—including motion and vertical/horizontal parallax. The

disadvantages of volumetric displays are that they are expensive, difficult to design and generate content. They are only used in specialised environments in the medical and military industries [34].

Figure 2.15: The Sony Ray Modeler autostereoscopic display projects 360 individual slices of an image onto an array of special LEDs, providing a volumetric display you can walk around and view from any angle – without glasses [33]

2.5.4 Holographic 3D Displays

Unlike volumetric displays, holographic 3D displays are projected onto a flat screen. They use the diffraction of light rays, using the inference patterns in the light waves, to generate a 3D image, recorded on a photographic film [35]. It is a well-recognised technology that, like volumetric, offer full parallax displays. The simplest method for recording a holographic image is to divide the laser beam into two separate beams. One of the laser beams passes directly through a lens spreading through the film plate, while

the other beam is reflected off the object before it is spread on the film. The reflected beam arrives with a slight delay. A holographic image is created when the two beams interfere on the film plate, creating an identical model of the original object with a 360 degree viewing angle.

However, the requirements for creating holographic 3D display/videos, including a coherent light source, a height level of mechanical stability, and dark room conditions, make it an impractical 3D display system. While there are successful applications for using holographic imagery on consumer products for security purposes [36], its practicality in using it for 3D displays is limited.

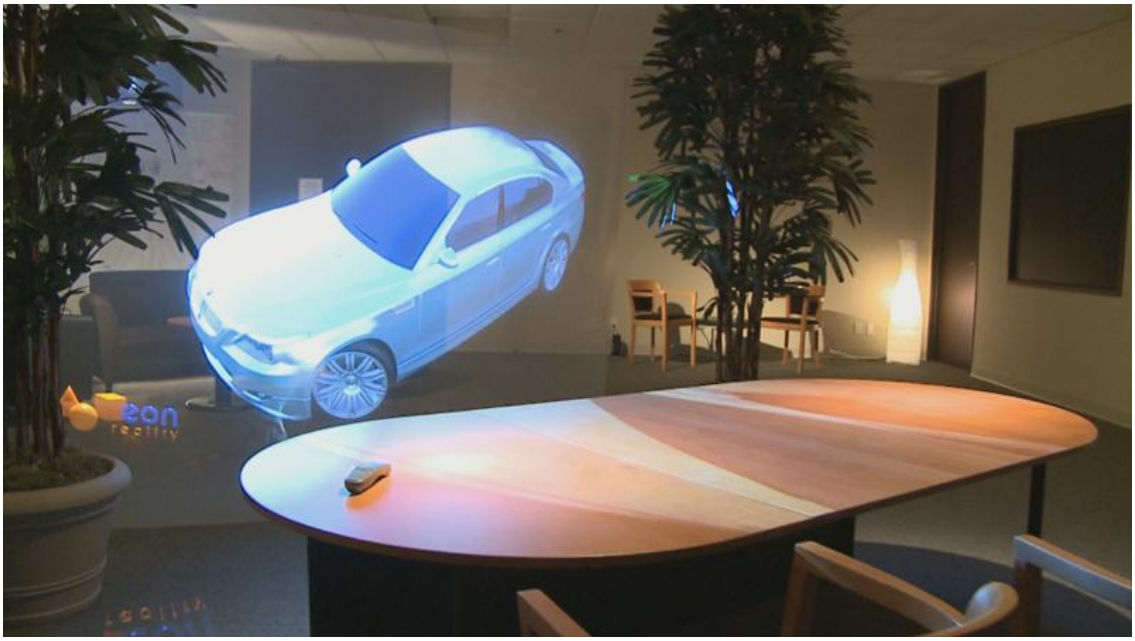


Figure 2.16: A demo picture of the Holographic display built by EON Interact, the EON Holographic I [41]

There are a range of holographic display systems developed [37][38][39][40][41]. An example of holographic display, shown in the Figure 2.16, uses holographic

geometrical principles to create holographic 3D vision. EON Holographic I, developed by EON Interact, is where 2D images are projected onto a 3D screen to create the illusion of a 3D system [41]. This system is perhaps the most practical because of the need for a single camera lens for filming and a single projector for playback. But even this is not a system for the general public, reserved for use during special demonstrations. Creating holographic displays require many other systems that exploit the holographic geometric principles, which have been developed by companies such as Hologaficka, Innovistion Labs, and viZoo to name just a few.

2.5.5 Holoscopic 3D Displays

Holoscopic 3D (H3D) display technology [42] is similar to the holographic 3D display technique with one key difference--it does not need coherent light source and confined dark spaces. Holoscopic imaging technology is also known as Integral Imaging technology. H3D uses microlenses to record and display the 3D scene using light intensity [43]. One of the advantages of integral imaging is the ability to refocus on different parts of the image after the image has been taken, commonly referred to as post processing [44].

It is worth mentioning that colleagues at Brunel University working on the 3DVIVANT research project have been actively contributing to this area [42][45][43][46]. One of their key contributions has been the deployment of a H3D camera prototype which builds on a 2D camera to capture holoscopic images as shown in Figure 2.17.

Integral imaging, despite its obvious advantages, is still an experimental technology. It is outside the scope of our research but interested readers can refer to

[47][48][49][50][51][52][53][54] for a detailed understanding. The references provided are not extensive.

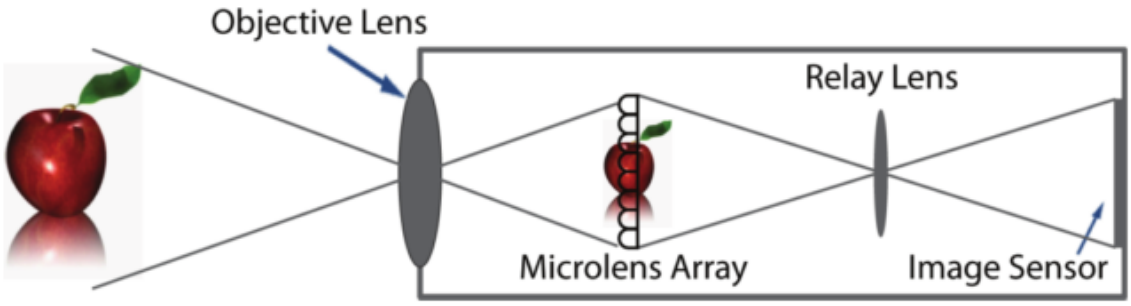


Figure 2.17: Illustration of holoscopic 3D camera design with a relay and objective lens [42]

2.5.6 Auto-stereoscopic and Multi-view 3D Displays

Auto-stereoscopic 3D displays (AS3D) pursue natural viewing as they do not require glasses, but still employ stereoscopic technology. They instead use space-division multiplexing to display 3D videos, directing light to the left and the right eyes through the use of parallax barriers or lenticular lenses. Because there is no need for the viewers to wear special glasses to perceive 3D depth, this approach is considered to be more convenient and practical [55]. AS3D can have two views or more--only using two views does not provide any motion-parallax because it is still a binocular display system.

The most promising display currently available is the multi-view 3D displays. They build on the auto-stereoscopic concept but offer more views, which makes it able to have motion-parallax.

Multiview 3D (M3D) displays [14] try to mimic the real world view still using stereoscopic technique, by using two views at each viewpoint to perceive 3D depth and offer motion parallax as the eyes move from one viewpoint to another. By doing so, it tries to mimic the real world, with multiple prospective views to immerse into. It of course cannot provide infinite views as we would in the real world, but even the finite numbers that it can provide takes a step closer to immersive realistic 3D perception.

In [56], the concept was first proposed to use the parallax barriers technology that would provide multiple views or windows to viewer by making two adjacent views at one time, which was later engineered by Frederic Ive in 1901. This gives the user the ability to perceive 3D without the need to wear glasses and offer motion parallax when the viewer changes or moves their head, where a different pair of stereoscopic images offer a different angle as it would in real world. The parallax barrier technology is depicted in Figure 2.18. Parallax barriers, when placed in front of the display, channel the left and the right views to the correct eye through the use of occlusion and stereo pair images placed in a precise grid.

Parallax barriers technology as depicted in Figure 2.18 was a breakthrough in creating realistic 3D perceptive technology but it had several disadvantages, the first being viewing distance limitation. The viewer can only perceive high quality 3D within specific viewing distance as shown in Figure 2.18. If the viewer is not within the recommended viewing zone, the viewer can no longer perceive 3D depth [31]. The second problem is visible dark areas caused by the barriers between pinholes that are put in place to separate the view. In addition, the motion parallax offered by parallax barriers if viewed from outside the supported viewing area can cause motion sickness

and eye fatigue due to the ghosting effect that is caused by the cross-talk between the images.

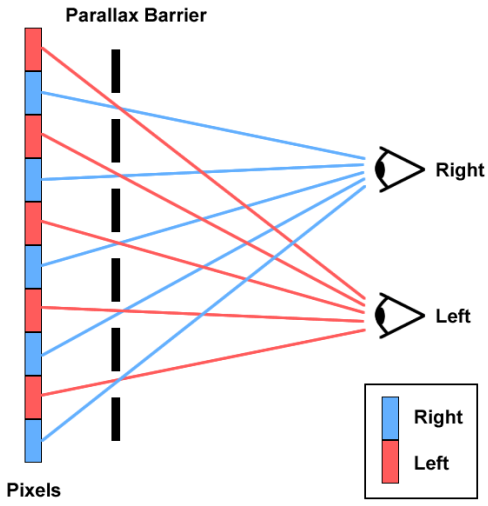


Figure 2.18: The concept of lenticular sheet to display multiple views in multiview 3D technology [31]

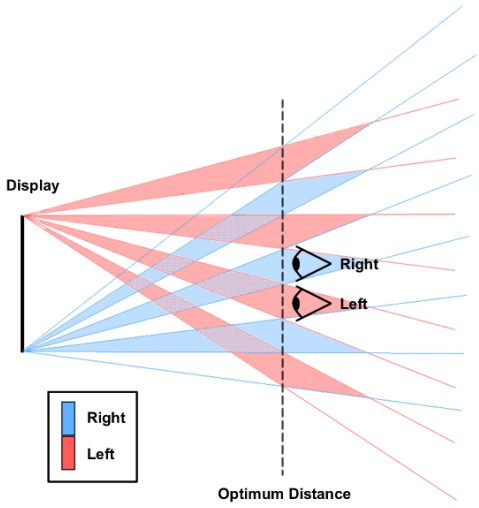


Figure 2.19: The problem with lenticular sheets is the viewing distance from the display. If the viewer is not in the optimum viewing zone they will see a distorted image, causing discomfort [31]

A solution was proposed in [57] which solved the motion parallax issue, where a head-tracking system would monitor the viewer's movements and adjust the viewing angle accordingly but this system could not support multiple viewers, making it only applicable to mobile devices used by individuals, rather than TVs that are built for communal use. Besides, replacing glasses with a head tracking device is counterproductive both in terms of comfort and monetary cost savings.

Lenticular lenses technology was proposed in [58] to solve the problem of the dark areas between pinholes and improve the quality of the image. The concept of lenticular lenses is very similar to parallax barriers with one significant difference--parallax barriers use the occlusion of light to separate the left and the right view whereas lenticular lens technology use refraction of light, which improves the brightness that the lenticular barriers reduced to half through occlusion.

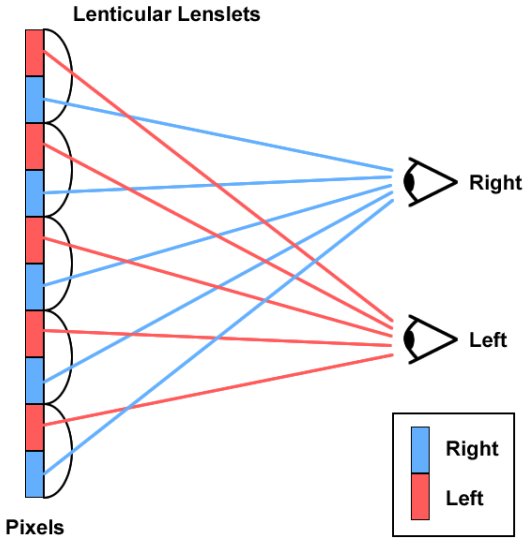


Figure 2.20: Principle of multiview 3D imaging based on parallax barriers and lenticular technology [31]

Although lenticular lens technology improved the image quality, it brought with it the moiré effect caused by the overlapping of the LCD pixel sheet patterns with the lenticular lens patterns [59]. Solving the lenticular lens sheet issue is an active area of research with several solutions offered in [60][61][62] which tilts the pixels to reduce the moiré effect. It is outside the focus of this research to explain here but interested readers can refer to the above mentioned references for details.

M3D display is a promising technology with bright prospects [63] that could increase the public demand for 3D television and displays in general because of its proximity to real world 3D perception. It is also less complex and cheaper to produce than its alternatives—holographic, volumetric and integral imaging based displays. The other major advantage is the content creation for multi-view is simpler. Real-time 2D to multiview content migration, offered by some displays such as Ultra-D [64], makes it even more attractive. Migrating from 2D to 3D in Multiview display technology is easier compared to volumetric, holographic and holoscopic displays because multiview video capturing uses the same cameras used for 2D video capturing. The only difference is there are more cameras used in Multiview video capture.

2.6 Delivery of 3D Videos

In the interconnected world that we live in today, the success of 3D systems must have video delivery as a core part of the 3D eco-system. Video transmission is not a new concept. The success of video sharing sites such as YouTube and Vimeo demonstrate the demand for video consumption. This, however, has created a problem for the internet backbone-networks with an estimated 30 to 40% of the bandwidth consumed

by video delivery [65]. 3D videos as described in this chapter require more data to represent them than their 2D counterparts. With the demand of multi-view auto-stereoscopic 3D videos increasing, each delivery system must find suitable solutions to handle the increase in data transmission. This section explores the different transmission channels and the concepts employed to make them work.

2.6.1 3DTV broadcasting

3D TV has been an active research area and perhaps one of the most developed area in 3D video delivery using Digital Video Broadcasting-Terrestrial (DVB-T), Terrestrial Digital Multimedia Broadcasting (T-DMB), or Digital video Broadcasting-Handheld (DVB-H), as the underlying technologies. Several approaches based on 2D TV broadcast technologies have been proposed and used. Considerable work has been done by the 3DTV project [66], supported by the European Union Framework Programs for Research and Technological Development, as part of FP6 and FP7. One of the advantages of 3D TV implementation is that existing broadband protocols and compression standards do not need to be changed for transmission.

In [67] a broadcasting technique has been using DMB-T. Two other solutions proposed for 3D content broadcast based on DVB-H using H.264/AVC or its MVC extensions are proposed in [68][69]. Both suggest the use of simulcast encoding, the first with side-by-side representation and the second as full image representation. The European ATTEST [70] project also proposed a 3DTV broadcasting approach using DVB-T transmitting video plus depth, under the pretext that the additional overhead presented by the depth information is considerably less than transmitting a second view.

In [71] Hur et al. present two new systems based on DMB and DVB-H for mobile 3D video transmission. In their experiments, they used multi-view video coding (H.264/MVC) to encode the stereo pair in a frame-compatible format, horizontally halved in resolution, before transmission over Real-Time Transport Protocol (RTP). The left view is used as the main view and in cases of error displayed in 2D for backward compatibility purposes, as MVC is built on the H.264/AVC standard. The left and the right views are transmitted during different time slices or bursts to increase error resilience. DVB-T modulator is then used to transmit the video in MPEG-2 Transport Stream (TS) packets.

The challenges with transmitting 3D over terrestrial TV networks are threefold; first, the channel bandwidth is restricted to 6MHz and the transmission bitrate to 19.4 Mbps [71]. While most 2DTV broadcasts allocate less than 17.5 Mbps for transmitting a video, having to split that between the stereo pair of images required for 3D transmission means compromising on the quality of the video, especially in cases of backward compatibility with 2D. In other words, the 2D video quality of the primary view in HD resolution will suffer from the reduced bitrate.

Most 3DTV terrestrial transmissions use the frame compatibility. This means halving the resolution of each video to fit into the allocated bandwidth. Since most 2DTVs use MPEG-2 as the coding standard, the primary view will have to be encoded using MPEG-2 and the right view using H.264 video codec respectively and use different bitrate to each view [71]. Both degrade the HD quality of the primary view which unfortunately also means the quality of the 3D video is also affected. The second problem is maintaining a timing constraint between the two videos that have been

multiplexed into one stream, raising the issue of buffer management at the receiver. The third problem is that current HDMI 1.4 products available support 1920x1080p at 30Hz whereas the stereo video requires 60Hz for each view.

2.6.2 3D Video Transmission over the Internet

Transmission of 3D videos over the internet faces different challenges. The bandwidth is one of the biggest concerns to date, as well as the heterogeneous nature of IP networks, ranging from Ethernet (wired) to 4G (wireless), each with a different set of preferences and architectures. There are a range of protocols that can be used to transmit 3D videos over the internet such as RTP, UDP, and HTTP [72][73][74][75].

A number of researchers have already addressed 3D video communication. In [7], a system is shown that uses JPEG still-image coding to encode stereoscopic views. They are then streamed in a single RTP session for shutter-glasses display.

In [74], a system using stereo modified H.264/AVC has been proposed using RTP protocol for transmission. The focus of this work, however, is mostly on coding optimisation in stereoscopic 3D videos. Several researchers have also suggested peer-to-peer architecture for multiview video delivery.

Transmission of 3D videos is still an active research area and the focus of Chapter 4 and 5 in this thesis. It will be discussed in more detail. This section just provides an introduction to the topic.

2.7 Challenges in 3D Eco-System

There are several challenges still facing the 3D systems ranging from content creation, error resilience and concealment, heterogeneity, achieving true free-view-point TV, quality of experience. The focus in this thesis is the end-to-end optimisation of multi-view videos before, during and after transmission. The challenges described below therefore relate to multi-view videos.

3D multiview videos are captured using 2D video cameras, usually with identical specifications. The number of views required is directly proportionate to the number of cameras used. Multiview displays use a different number of views, so the number of cameras must be adjusted to the display. Aliscopy multiview 3D display has eight pixels per lens under its lenticular lenses which means the number of views it needs is eight [76]. That is eight 2D cameras capturing the same view from slightly different angles.

Display manufacturers have tried to overcome this problem by including a 3D rendering engine in the displays to generate multiview videos from 2D plus depth images [76]. The other way to overcome this problem is by generating a depth map from 2D images [77]. But these solutions do not offer real multiview 3D experience because the images created from 2D plus depth or stereoscopic image pairs cannot offer enough angular visual information as multiple cameras. But capturing multiview 3D videos with multiple 3D cameras requires complicated camera calibration and synchronisation [78].

3D video communication solutions must consider the entire pipeline, starting with optimisation at the video source to the end display and transmission optimisation. Multi-view offers the most compelling solution for 3D videos with motion parallax and freedom from wearing headgear for 3D video perception. Optimising multi-view video for delivery and display could increase the demand for true 3D in the consumer market.

2.8 Conclusions

An overview of the trends in 3D video technological advancements is presented in this chapter. The underlying techniques in 3D video perceptions are discussed followed by the way they are represented in videos. Different representations affect the video perception depending on the compromises made to represent them. The most ideal representation is where all views are used with full resolution. This makes for the best 3D perception but increases the amount of data that needs to be recorded proportionate to the number of views captured.

3D displays correlating to the 3D video representation are also discussed, including those that are still under current investigation in research. The most promising display that could increase the uptake of the 3D video displays with the general public is the Multiview Autostereoscopic 3D displays. One challenge with multiview displays is the requirement to capture the different views using 2D video cameras, which need to be synchronised and calibrated. But the biggest challenge is to optimise them for delivery over the internet and display.

Compression of 3D videos is also introduced in this chapter, paving the way for a more detailed analysis in Chapter 3. Since compression of the 3D videos depend on the number of views, the best compression technique would have to exploit the interview redundancies and provide an optimum solution for delivery over heterogeneous networks.

Transmission of 3D videos is discussed. Transmission techniques used in 3DTV broadcast, which is the most mature and well-researched delivery mechanism, is

analysed. The future of any 3D video transmission is foreseen to be over the internet. Transmission over the internet is introduced and the transmission methods briefly explained. They will be discussed in more details in Chapter 4 and 5.

The next chapter presents a literature survey of the video compression and the state of the art in multi-view video coding. A comparison of the traditional compression technique—simulcast—and the state of the art codec H.264/MVC is presented. The study is focused on understanding the effect of exploiting inter-view redundancies on video quality and whether any coding efficiency is achieved as claimed by the codec. The chapter compares the coding efficiency with the loss in video quality to evaluate whether the decrease in the resultant encoding bitrate is worth the loss in video quality.

Chapter 3: Compression Techniques in 3D Videos

Compression of video has long been exploited in 2D videos. The most commonly used and currently state of the art encoder/decoder is H.264/MPEG-4 Part 10 or Advanced Video Coding (AVC). It efficiently exploits the redundancies between consecutive frames to reduce the amount of data for storage or transmission while maintaining the quality of service. This chapter investigates the compression of 3D video and its evolution from the 2D predecessor. A detailed survey of the relevant compression techniques is presented. The benefits of exploiting inter-view redundancies are demonstrated through experimental results. Stereoscopic video signals are compressed using the 3D feature of the ITU-T Video Coding Standard H.264/AVC. The work analyses the coding of 3D video with regards to both perceptual video quality and bitrate generation, with the equivalent 2D signal (single view) setting a benchmark. The perceptual quality is demonstrated objectively in order to determine the effect of the additional view on the perceptual 3D video quality. The bitrate analysis aims at estimating the range of bitrate increase that is expected when two views are compressed compared to single view compression. The objective results shown in the chapter for both the quality (PSNR) and bitrate metrics are produced from a series of experiments run under different encoding scenarios in order to tabulate average figures in a conclusive and reader friendly manner. Experimental results show that exploiting inter-view redundancies in stereoscopic videos increases coding efficiency by 24% compared with simulcast—where each view is independently encoded.

3.1 Introduction

3D video has seen considerable attention from the research community over the past decade. The interest in 3D videos has been fuelled by the fascination of the general public created by 3D movies. Today there are sports channels that broadcast 3D content and the expectation is to provide similar services over IP networks. The most common type of 3D content is the stereoscopic video which generates two views for the left and right eyes. The Human Visual System fuses the two images to form perceptive 3D depth. The images are filtered by either a pair of glasses of different types such as anaglyph glasses and shutter glasses--which synchronize with the display to feed different images to the left and right eyes—or by a display system to create an auto-stereoscopic video through parallax barriers or lenticular lenses. In the case of the latter, the viewer does not need to wear glasses to perceive a 3D image. The images are separated by the mentioned optical elements to be viewed by each eye separately. Chapter 2 (section 2.5) discusses 3D display technologies in detail. Interested readers are also referred to [13] where the authors have discussed 3D displays and communications challenges in details. The focus of this chapter is to introduce compression techniques in 3D videos using the H.264/AVC and the challenges it faces from bandwidth restrictions.

H.264 Advanced Video Coding (AVC) is the de facto standard from the Joint Video Team (JVT) for video coding. It is defined by ITU-T recommendation for advanced video coding for generic audiovisual services [79]. In addition to having the ability to encode single view video streams, it has been extended to include the increasing demand from 3D video. The stereo and multiview video coding extension is described

in Annex-H of the ITU-T recommendation [79]. It deals with the increase in encoding complexity and the additional bandwidth requirement. All stereo and multiview streams are backward compatible with 2D decoders [80]. This makes it the most logical encoder to use for experiments done as part of this research.

3.2 State-of-the-Art in Video Codecs

One of the most widely used codecs, which enabled digital television systems worldwide [79], is the MPEG-2 video coding standard (also known as ITU-T H.262). It was standardised in the 1992. In addition to being used for terrestrial television transmissions, it is also used for the compression of both standard definition (SD) and high definition (HD) videos for storage of videos on DVD and transmission over satellite and cable [85].

However, as the need for more efficiency in coding came about in the internet age, where Cable modems, DSL, and UMTS had a much lower data rate than broadcast channels, the H.262 codec evolved through a series of developments to H.263 and later to H.264/MPEG-4 advanced video coding (AVC) [85]. With each improvement the coding efficiency was improved, effectively increasing the capacity of the transmission channels. The additional capacity meant that existing transmission channels can now transmit higher quality videos.

H.264/AVC, developed by the Joint Video Team of ITU-T Video coding Experts Group (VCEG) and the ISO/IEC Moving Pictures Experts Group (MPEG), is still considered the state-of-the-art codec for 2D video. It is being used for broadcast over

cable, satellite, and the internet. It is also broadly used in consumer electronics, from video cameras and blue-ray discs to digital TV set-top boxes [86].

Given the success, efficiency, and widespread adoptability of H.264/AVC, JVT extended it for 3D videos—the Multiview Video Coding (MVC) extension of H.264/AVC (commonly referred to as (H.264/MVC) [79]. The MVC extension has two profiles for encoding 3D videos, stereo high profile and multi-view high profile [80]. MVC exploits the interview redundancies between views and is also backward compatible with AVC; therefore, any AVC decoder can decode the base view of an MVC encoded bitstream.

A new standard, the High Efficiency Video Coding (HEVC) or H.265, recently introduced in January 2013 by the Joint Collaborative Team on Video Coding (JCT-VC), [87] aims to address new services such as 4kHD and multiview video capture and display. HEVC is specifically designed to focus on increased video resolution and use of parallel processing [87]. In [14] D Milovanovic and Z Bojkovic compared H.264/AVC with H.265/HEVC and shown that HEVC provides 59.35% average compression efficiency compared to AVC but the coding time exceeds ~70% on average compared to AVC [88]. Since HEVC is still a new codec and more focused on the HD videos, the focus of this research is on the MVC extension of AVC.

3.3 H.264/Multi-view Video Coding (MVC)

MVC, standardised in 2009, covers a wide range of 3D video applications including 3D video streaming, free-viewpoint video as well as 3DTV [83]. It is an extension of the AVC standard. This makes it inherently backward compatible with AVC which serves

as the base-view that can be decoded independently in the absence of the MVC decoder [80]. Any additional views are referred to as enhancement views and are typically coded using interview prediction within the same bitstream [84].

It is beyond the scope of this research to discuss the H.264/AVC standard but those features of the AVC that are relevant to MVC and are also relevant to this research are discussed below. Interested readers can find details of the AVC in the ITU-T Recommendations for H.264 – Advanced Video coding for generic audio-visual service document [79] and for an overview refer to [85]. Details about the MVC extension can also be found in the annex-H of [79] and an overview presented in [80][84][18].

AVC has two key components—the video coding layer (VCL) and the network abstraction layer (NAL). They work together to serve a heterogeneous client base. The VCL represents all the video content. The NAL encapsulates the VCL data to provide meaningful information to the decoding client and transmission channels for effective interpolation, through the information in each NAL unit header. NAL and VCL are described in details in the following sections.

3.3.1 Video Coding Layer (VCL)

AVC uses a hybrid block based video coding approach, with an increased flexibility and adaptability compared to its predecessors. Pictures are divided into slices, which in turn are further divided in to macroblocks. Slices are independent of each other and hence can be parsed without affecting other slices in the frame. [80]

Macroblocks cover an area of 16x16 luma samples. In 4:2:0 chroma sampling format videos the macroblocks cover 8x8 of each of the two chroma components. Macroblocks

are spatially or temporally predicted with three main slice coding types, intrapicture coding (I), predictive coding (P) and bipredictive coding (B) [79][80].

I slices are spatially predicted from neighbouring regions. P slices can be predicted from both intrapicture and interpicture regions with only one signal for each predicted region. B slices can be predicted from intrapicture, interpicture and interpicture biprediction. It uses two prediction signals combined to form a weighted average from both regions. [80]

One of the problems with block based coding is the blocking artifacts. AVC employs an adaptive deblocking filter in the motion compensated interpicture prediction loop [80] to reduce the blocking artifacts.

The coded slices are then transformed using Hadamard transform, quantised using a uniform reconstruction quantizers and entropy coded using either context-based adaptive variable-length coding (CAVLC) or context-based adaptive binary arithmetic coding (CABAC). CABAC uses a more sophisticated mechanism for statistical dependencies which is reported to have a bit rate saving of 10 to 15% savings compared to CAVLC [80].

In AVC, the picture order of coding and display are completely decoupled [80]. Any picture can be independently used as a reference picture for motion compensated prediction of future pictures. The decoded picture buffer (DPB) can be adaptively controlled by memory management control operation (MMCO) commands [80]. Reference picture list modification (RPLM) commands can be used to arbitrarily

construct a reference picture list for P and B slices from the pictures available in P or B slices [80].

3.3.2 Network Abstraction Layer (NAL)

AVC coded video bitstream is organised in NAL units (NALU) [18]. NAL units are packets that contain an integer number of bytes. NALUs act as an interface for the transport and system layers [84]. AVC has two types of NALUs identified by the 1-B indicator as either a video coding layer (VCL) NAL unit or a non-VCL NAL unit, with a variable payload. VCL NAL units contain coded video data whereas the non-VCL NAL units carry information about the decoding process settings such as picture parameter sets (PPS), sequence parameter sets (SPS), and supplemental enhancement information (SEI) messages [80][84].

Although, non-VCL NALUs do not contain coded video, they carry information that helps the decoder with bitstream manipulation and display. Although they do not affect the core decoding process, they usually contain infrequently changing information; SEI messages, for example, contain information for the decoder which can be used for bitstream manipulation and/or display [80]. SPS contains video dependency information, a key feature of MVC, which helps signalling-aware media gateways to construct view dependency trees [18].

An independently coded sequence contains a set of consecutive access units, which are a set of consecutive NALUs that belong to a single coded picture, with certain properties and their associated parameter sets [80]. Every independently decodable video sequence begins with an instantaneous decoding refresh (IDR) access unit. The

IDR access unit tells the decoder that it and all the following access units can be independently decoded without the need for any of its preceding pictures [80].

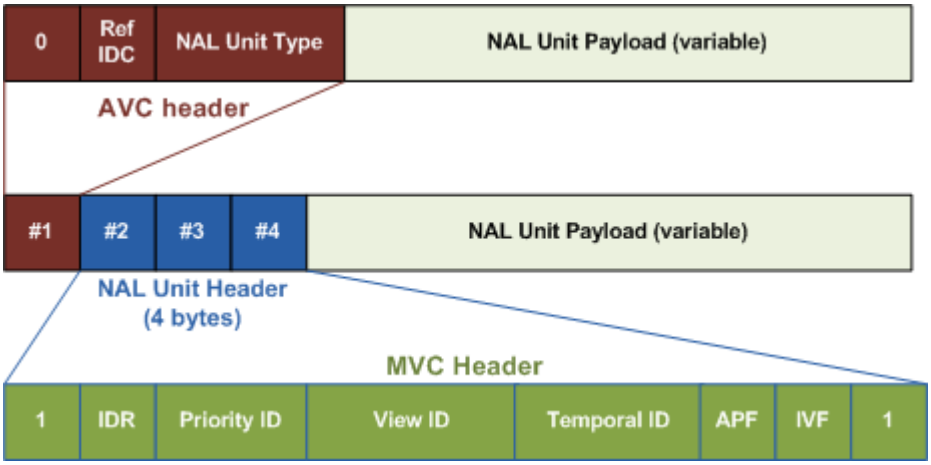


Figure 3.1: MVC NALU interface [84]

MVC bitstreams are backward compatible with AVC. The NALU structure is extended to allow for the additional information relating to the enhancement views to be added to the NALU header. The 1-byte AVC header is extended by 3 bytes, as shown in Figure 3.1, and a new NALU type is added (NALU type = 20) for MVC extension [84]. If an MVC bitstream that contains NALU of the new type is seen by legacy AVC decoders, they ignore it and only decode the base view (NALU type = 14). Note that the 3-byte extension is not exclusive to MVC. It is also used for Scalable Video Coding (SVC) extension of AVC.

The NALU type, therefore, plays a key role in identifying its content. Each of the new header syntax types namely, idr_flag, priority_id, view_id, temporal_id, anchor_pic_flag, inter_view_flag, help perform different functions. For instance, the

anchor picture flag can be used as random access points if the flag is on [18]. A combination of these flags can be used for random access and view switching.

View switching, as the name suggests, is to change the view(s) being displayed to the user. This could be a pair of stereoscopic views, a single view, or multiple views. View switching is an important application for free-viewpoint video [18].

Random Access in MVC refers to starting the video at any point other than the start. Digital 2D videos have made this a common and expected feature in videos. In MVC this is more complicated because of the additional views.

MVC makes random access and view switching possible with its bitstream extraction features available in the NALU header as described above. Two important syntax elements named earlier, view_id and temporal_id, help with the extraction of the target views and required adaptation. Temporal_id in an MVC bitstream indicates the frame rate [18]. The priority_id flag is usually used for a simple one-path bitstream adaptation process. It is set by the encoder and can be used by media gateways to discard NAL units higher than a defined value [18].

IDR frames are the best place for random access points. In MVC, only the base view has IDR frames (NALU type = 5), while the preceding views are dependent on the base view. However, the IDR flag, known as view-IDR or V-IDR, in the extended MVC header is turned on when the base view frame is IDR. V-IDR pictures only depend on the views in the same access unit. MVC employs time-first coding, which means access units contain all the NALUs that belong to a certain time instance, hence the IDR picture in the base view can continue the random access points [18].

3.3.3 Inter-view Redundancies

MVC exploits the redundancies between views in a more comprehensive manner. Figure 3.2 shows the dependency structure between the views, assuming there are eight views with a Group of Picture (GOP) size of eight. The first view in an MVC encoded video is independently decodable, to make it backward compatible with AVC. The other views depend on the first view as a reference to increase the encoding efficiency. Coding efficiency in this context is referred to a smaller file size, decreasing the amount of storage space required and a reduced amount of data for transmission.

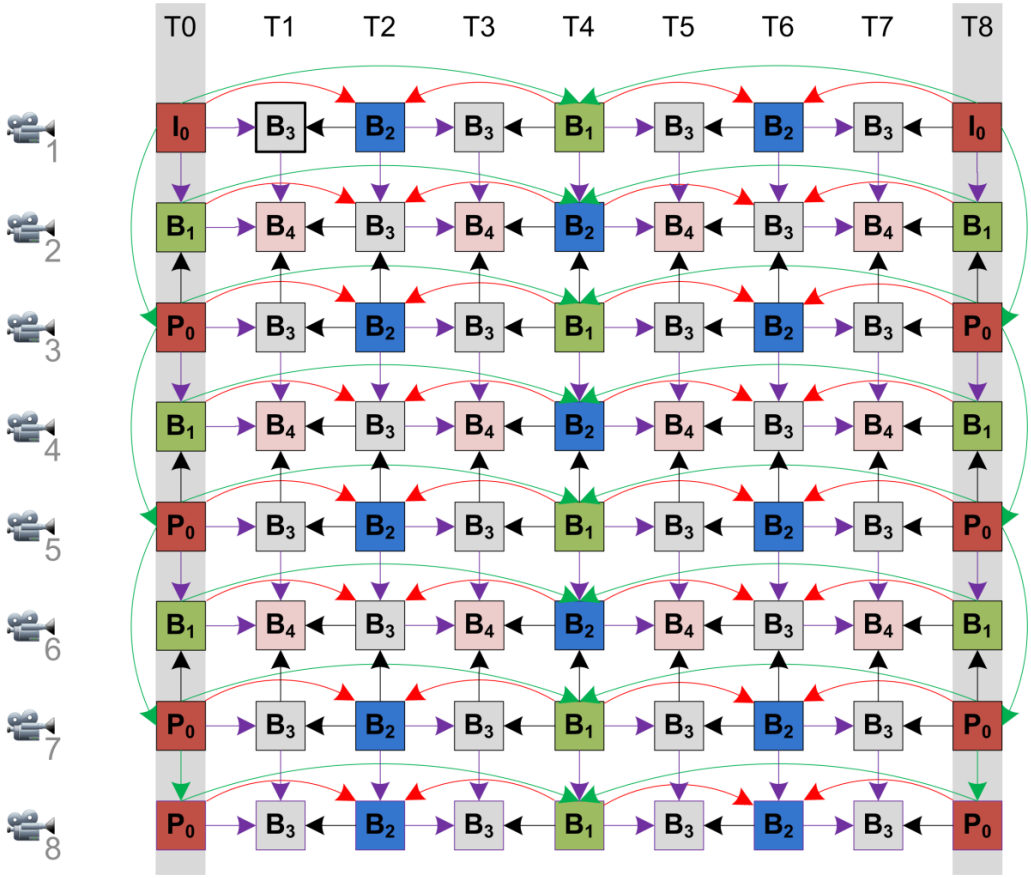


Figure 3.2: Prediction structure of H.264/MVC. Note the inter-view dependency is much more comprehensive than in the stereo profile of H.264/AVC

Inter-view prediction structure is similar to the temporal prediction that is common to 2D videos, the only difference being the prediction happening between frames of different views. Temporal prediction within each view still happens. As shown in Figure 3.2, MVC employs both the traditional temporal prediction and a new type of prediction that happens between adjacent views called inter-view prediction. One of the disadvantages of such a design is that transmission errors in the reference view can propagate exponentially in subsequent views. This is discussed in more detail in Chapter 4, 5 and 6.

3.4 Compression Options for Stereo Video

Stereo videos have two views, Left and Right, which doubles the encoding complexity and bandwidth requirement for transmission. There are two techniques that can be used to encode the two views. The most basic is to treat each video as separate videos and encode them separately as depicted in Figure 3.3. This technique is called simulcast, as each view is independently encoded.

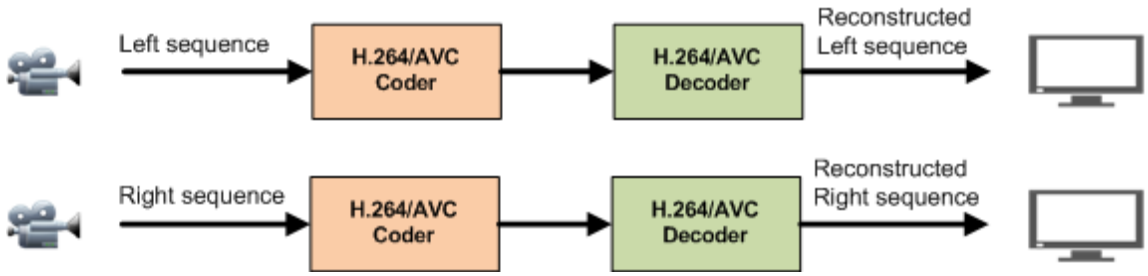


Figure 3.3: Encoding the left and right views as independent views

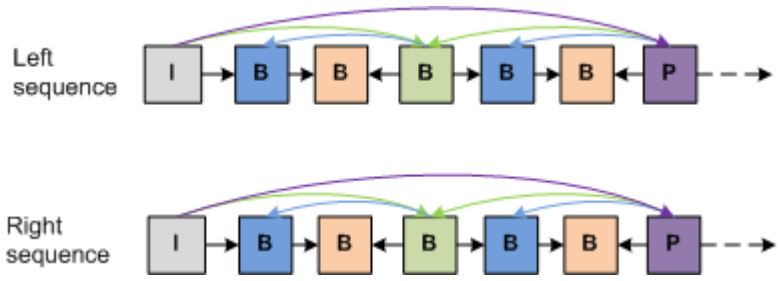


Figure 3.4: Example of temporal prediction in each view

Figure 3.4 shows prediction when the two views are encoded independently. While this method has its benefits, it does not exploit the redundancies that are preset in between the views. Figure 3.5 shows the stereo encoding where the two views are incorporated in one transmission stream for better encoding. While this increases the coding complexity, it exploits the similarities that are present between the two views. Figure 3.6 shows the interview prediction when the two views are incorporated. As depicted by Figure 3.6, the interview prediction produces a more efficient bitstream.

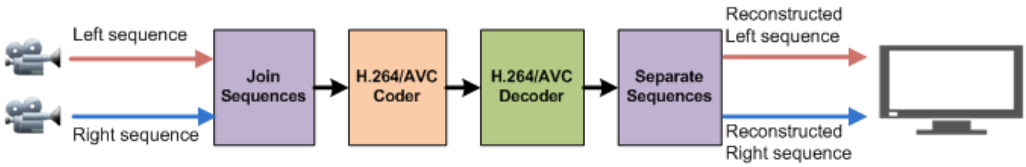


Figure 3.5: Incorporating the two views of a stereoscopic pair into one stream

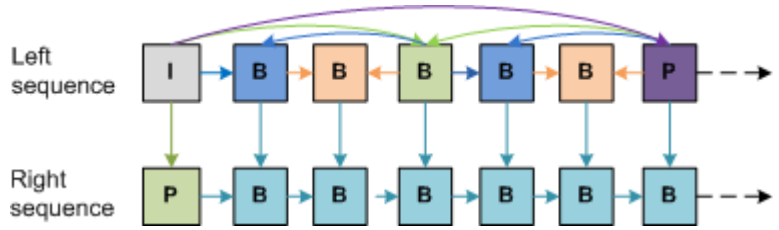


Figure 3.6: Example of temporal and interview prediction in the stereoscopic pair [80]

3.5 Comparison of Simulcast and MVC

Three standard videos namely Ballroom, Vassar, and Exit [81] were used to test the two scenarios. The left and right views are 10 cm apart. Figure 3.7 shows frame 63 of the ballroom sequence for the left and right views respectively.

The videos were put under different conditions by varying the quantization parameter (QP) to see how the different conditions affect the bitrate and PSNR of the videos. The QPs used for each video were 28, 31 and 34. The resolution of all three videos was 640x480 pixels. The number of frames encoded was 125 for each test at 25Hz, i.e. each sequence is 5-second in length.

The number of reference frames is set to 2, with hierarchical B-coding enabled and 7 B-coded frames in between two anchor frames. In total 27 tests were carried out. The benchmark used for the stereo pair and the independently encoded videos (simulcast) is the single view.


(a)

(b)

Figure 3.7: Frame 63 of the Ballroom sequence of the left (a) and right view (b) respectively

3.6 Experimental Results

The test results shown in Table 3.1 illustrate bitrate savings between ~3% to ~24% could be achieved by using stereo profile which exploits interview redundancies. From Table 3.2, it is evident that this additional bitrate saving has a negligible effect on the quality of the reconstructed view. More savings can be achieved if both views are halved in spatial resolution, however, that would degrade the quality of the video.

		Bitrate (kbits/s) @ 25.00 Hz								
		Single (view 0)			Simulcast (view 0 + view 1)			Stereoscopic		
	QP	28	31	34	28	31	34	28	31	34
Sequences	Ballroom	1088.33	727.7	484.44	2218.65	1479.3	985.36	2139.78	1415.78	938.61
	% Diff.							3.55%	4.29%	4.74%
	Vassar	289.99	142.15	79.53	582.59	280.03	155.48	447.98	234.05	137.15
	% Diff.							23.11%	16.42%	11.79%
	Exit	336.4	207.39	133.2	692.58	422.78	271.42	634.9	394.96	263.52
	% Diff.							8.33%	6.58%	2.91%

Table 3.1: Bitrate values of test sequences

		Bitrate (kbits/s) @ 25.00 Hz								
		Single (view 0)			Simulcast (view 0 + view 1)			Stereoscopic		
	QP	28	31	34	28	31	34	28	31	34
Sequences	Ballroom	38.53	37.08	35.53	38.49	37.06	35.53	38.55	37.14	35.63
	% Diff.				0.04	0.01	0.00	-0.06	-0.07	-0.09
	Vassar	38.21	37.19	36.14	38.20	37.22	36.18	37.92	36.96	35.92
	% Diff.				0.01	-0.03	-0.04	0.28	0.26	0.25
	Exit	40.00	38.91	37.68	39.95	38.88	37.67	39.70	38.60	37.37
	% Diff.				0.04	0.02	0.01	0.25	0.28	0.29

Table 3.2: Average PSNR values of test sequences

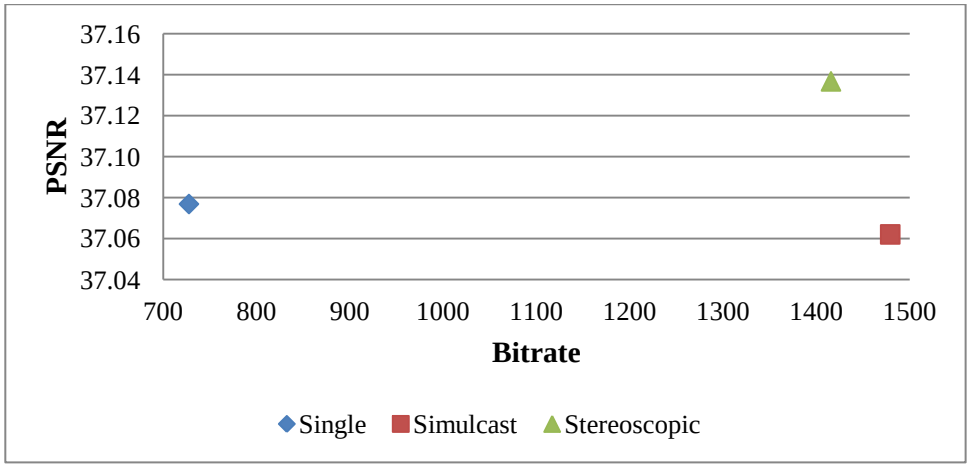
To explain these tables graphically, the QP is fixed at 31 for all three of the videos, Ballroom, Vassar and Exit to make the comparison between videos in terms of bitrate saving compared with quality (PSNR) loss. Results in Figure 3.8 show the bitrate of

single view, simulcast, and stereoscopic video encoding schemes compared to PSNR of the three video sequences. As it is anticipated, the bitrate generated by coding two views simulcast is about twice as much as in the single view case with very little quality loss. In the case of Ballroom, there is even a slight gain in quality. On the other hand, the usage of stereoscopic encoding could reduce the increase in bitrate to almost a half such as in Vassar at QP 28 as shown in Table 3.2. It is not as high using other QP values in other videos but Figure 3.8 clearly highlights the bitrate saving with the use of the stereoscopic profile of MVC compared with simulcast.

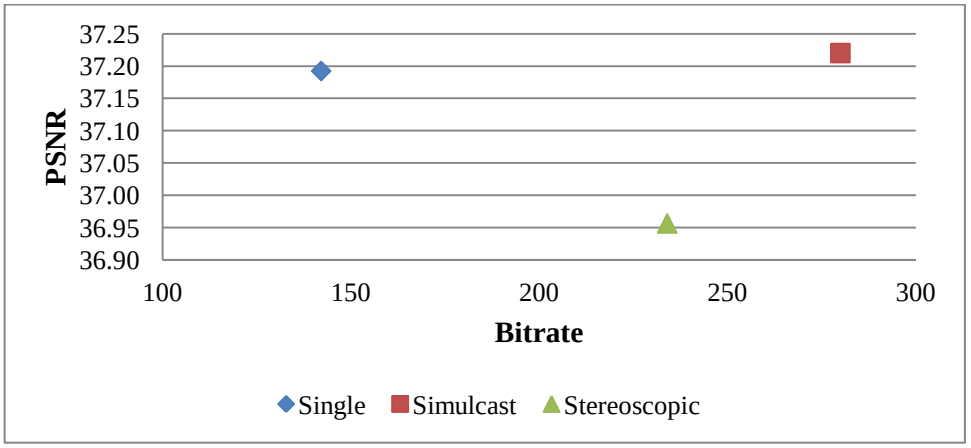
The improvement seen from the Ballroom sequence is not as substantial as the Vassar sequence as it is categorized as a complex sequence containing a lot of ballroom dance movements, resulting in less temporal redundancies between its frames. The Vassar sequence falls under the low activity category with stationary background and a moving object, which makes it of lesser complexity. The exit sequence is of a moderate complexity with considerable motions. In general, the amount of bitrate generated by stereoscopic coding is less than simulcast.

From Figure 3.8, it can also be seen that a higher QP value produces a lower bitrate for transmission or storage. The QP value is an indication of how much the spatial detail is retained. A small QP retains almost all the spatial information, whereas retention of that spatial detail is less when a higher QP value is applied.

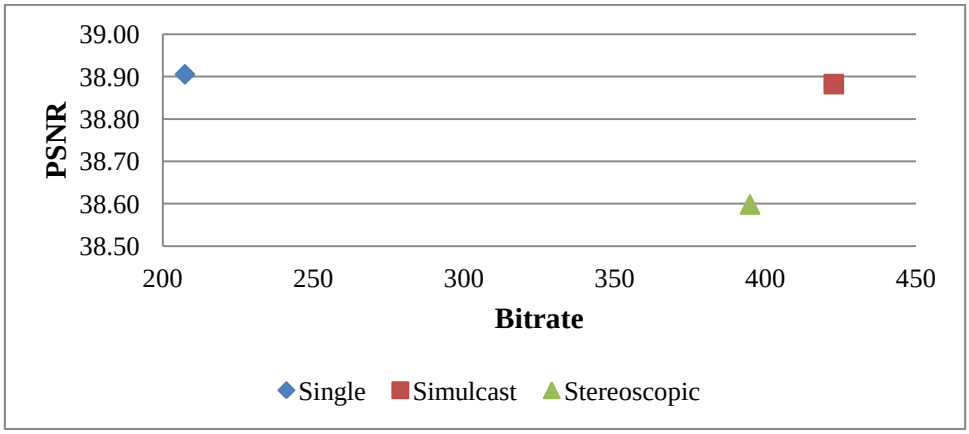
A good objective quality measurement of the reconstructed video is the PSNR, and that can be verified from Table 3.2.



(a)



(b)



(c)

Figure 3.8: Comparison of PSNR and bitrate when the QP is fixed at 31 in simulcast and stereo encoding with respect to single view encoding for (a) Ballroom, (b) Vassar, and (c) Exit sequences

PSNR for more than one view is calculated by averaging the individual PSNR values for each view. This approach can be used for both stereoscopic and multi-view scenarios. It can be observed that the simulcast coding achieves comparable video quality as compared to the single view coding. On the other hand, stereoscopic coding achieves a slightly lower quality. From our experimental results, the maximum quality degradation was computed to be -0.3dB obtained from the exit sequence (QP: 34), which is negligible.

3.7 Analysis and Discussion

Temporal redundancies in video compression have long been exploited for compression efficiency. In 3D videos the redundancies between views are also exploitable to further increase compression efficiency. The above results show two distinct methods for the compression of 3D videos.

The simulcast method, where each view is independently coded, only takes temporal redundancies into account. This method is the most clear-cut due to the fact that it can employ currently 2D encoders/decoders. It does, however, increase the amount of data to be transmitted or stored proportionate to the number of views. In stereoscopic 3D videos, for instance, it doubles the amount of compressed data.

Exploiting inter-view redundancies increases coding efficiency by up to 24% in stereoscopic videos compared with simulcast coding without any significant reduction in visual quality. The gains shown here are for full-resolution representation but it is not limited to full-resolution. These gains could also increase compression quality in frame compatible and 2D plus depth representation described in Chapter 2.3.

The codec used is H.264/AVC for both simulcast and stereoscopic coding techniques used in the above experiments. The stereoscopic and multi-view extensions of H.264/AVC are backward compatible. In the event the stream is decoded by 2D decoder, it will only decode the base view. In other words, encoding with the multi-view video coding extension of H.264 does not render the video un-decodable for legacy 2D AVC decoders.

3.8 Conclusions

Multi-view video coding enables auto-stereoscopic displays and Free View-point TV for the mass market. It is the most promising technology in 3D videos to date. But it comes with a large amount of data captured by multiple synchronised 2D cameras. The similarities between the views can be exploited for coding efficiency. Multi-view video coding (MVC) extension of the H.264/AVC standard is explained and its benefits highlighted.

A comparison of video compression on 3D video is presented with 2D video setting a benchmark. Two views are encoded with the stereo profile of the Advanced Video Coding (AVC) standard, first with treating each view as an independent view and then by exploiting the inter-view dependency. The results show that while there is no considerable quality degradation, the savings in bitrate can range from 3% to 24% in the test bed used in the experiments. The bitrate analysis presented in this work was to demonstrate the range of bitrate savings when two views are compressed in a stereoscopic encoding scheme as compared to simulcast compression. The stereoscopic coding scheme can be exploited further to reduce the bitrate by incorporating other techniques such as depth map estimation [82].

The results shown here demonstrate the importance of selecting the correct codec for robust compression efficiency. The stereo profile of the H.264/AVC where the redundancies between the views are exploited without any considerable effect on the quality of the video highlights how efficiency in 3D video can be achieved, which

provides the basis for using MVC as the standard codec for 3D videos. The next two chapters will focus on the transmission of 3D videos.

Chapter 4: 3D Video Delivery over HTTP

With the increased popularity of wireless communications and more advanced end points such as smart phones and tablet PCs, it is becoming increasingly demanding to deliver a variety of services, including 3D video services, which optimize the users' Quality of Experience (QoE). Some smart phones can now capture and play stereoscopic videos. Enabling these devices to deliver their content in 3D will open up new opportunities in the 3D video communications area. This chapter investigates an approach for delivering Multi-view Video using HTTP over wireless networks, with varied packet sizes depending on observed channel conditions. OPNET System-in-the-Loop module is used to transmit real video over a simulated network. It is commonly expected that small packet sizes will perform better in bursty wireless channels when the network is reported to be busy. Our results show that an optimum packet size for IP networks at the application level is 64 kB Smaller packets increase the transmission overhead in an already congested network and the communication between the physical and application layers incur additional delays. Larger packets at the application level also have an adverse effect on the transmission times. Packet size variation and its effect on transmission times is discussed in details.

4.1 Introduction

3D videos have a large amount of data that not all networks can handle. Wireless communication is bursty and lossy by nature. It is important to maintain the quality of video to a satisfactory level. Errors loss of a received packet can propagate to other frames and views. Since HTTP uses TCP to deliver, it guarantees the delivery of the packets but at the cost of extra time. This may not be the best option for video streaming but in cases where video quality is an important factor and time is not critical though an important factor in the user's quality of experience.

This chapter investigates the optimum packet size for an efficient video delivery. With H.264/MVC, in addition to exploiting the similarities between frames of the same view, it also exploits the similarities between inter-view frames [80] as discussed in Chapter 3 in detail. This greatly reduces the amount of traffic that needs to be delivered compared to simulcast transmission of independently coded streams [89]. Real video bit streams are transmitted via an OPNET simulated network using its System-in-the-Loop (SITL) module [90].

The quality of experience in a 3D video communication system can be affected by many things, including errors in signal processing, lack of required information, packet loss, and optical errors in display [13]. This can cause visual artifacts, visual discomfort and fatigue which impairs the user's QoE. This research focuses on the packet loss, and strives to ensure that while using a protocol that guarantees a packet arrival at the destination with 0% loss, the sender and receiver can adapt themselves to the changes in the channel to maintain constant delivery of packets and QoE. The rest of this

chapter is organised as follows: A brief overview of H.264/MVC is given, followed by the introduction of SITL before presenting the proposed system. The results from the simulation are discussed in light of the impact of using the proposed system. The chapter concludes the findings with recommendations for the future.

4.2 H.264/MVC

It is beyond the scope of this chapter to describe the Multi-View Video Coding (MVC) extension of the H.264/AVC (Advanced Video Coding) standard. A brief overview of the coding scheme and the packetisation of the Network Abstraction Layer Unit (NALU) is provided here to understand the delivery mechanism. For a comprehensive study of MVC, please refer to chapter 2 and 3. Readers can also refer to [80,89,91] for further study.

3D video is referred to many different visual formats. Stereoscopic video, where two slightly different views are presented to each eye, is one form of the most commonly used formats. Its usage may differ from one video to another, ranging from anaglyph to polarization or barrier parallax. The other format is the video plus depth. Multi-view is another format where more than two views are represented including in free-view point displays. This variation makes it difficult to build one transmission system that can apply to all. MVC can encode stereo and multi-view videos as well as video plus depth. This makes it an ideal encoder on which to build the proposed delivery system.

A standard MVC bitstream generated by JMVC 8.5 is used. The videos used for experiments are the original MERL videos available at [92]. The benchmarked sequences used are Ballroom, Exit and Vassar.

MVC has two types of NALUs identified by the 1-B indicator as either a video coding layer (VCL) NAL unit or a non-VCL NAL unit. VCL NAL units contain coded video data where as the non-VCL NAL units carry parameter sets and supplemental enhancement information (SEI) messages [80]. Although, non-VCL NALUs do not contain coded video, they carry information that helps the decoder with bitstream manipulation and display. For the purpose of the experiments for this research, these are important to get the quality of output required. The experiments in this research attempt to deliver the complete bitstream as generated by the MVC encoder without any modification.

4.3 System-in-the-Loop

System-in-the-Loop (SITL) is a module of OPNET simulator that allows for transmission of real packets over simulated networks. The proposed test bed is created using this module. SITL can be connected to communicate with two real end points communicating over simulated networks, real end point to communicate with a simulated node, or simulated nodes to communicate over a real network [93]. Here, the first scenario of real nodes communicating over simulated networks is used. The alternative to using SITL is to use video trace files that replicate the behaviour of the bitstream or transmit the video over a real network. Using trace files limits the ability to conduct subjective tests on the received video. Using SITL has the benefit of transmitting real videos over simulated networks between real client and server. Using video traces would also limit the objective test methods such PSNR comparisons. Although this is not applied in the work presented here, it is critical to future work and opens new opportunities for more in-depth analysis of the transmitted video. Figure 4.1

shows the communication of the simulation setup using SITL. A central computer with OPNET installation and access to the license server is used to create the simulated network shown in the test bed. Two computers are connected to the central computer, alternating as the video server and receiver.

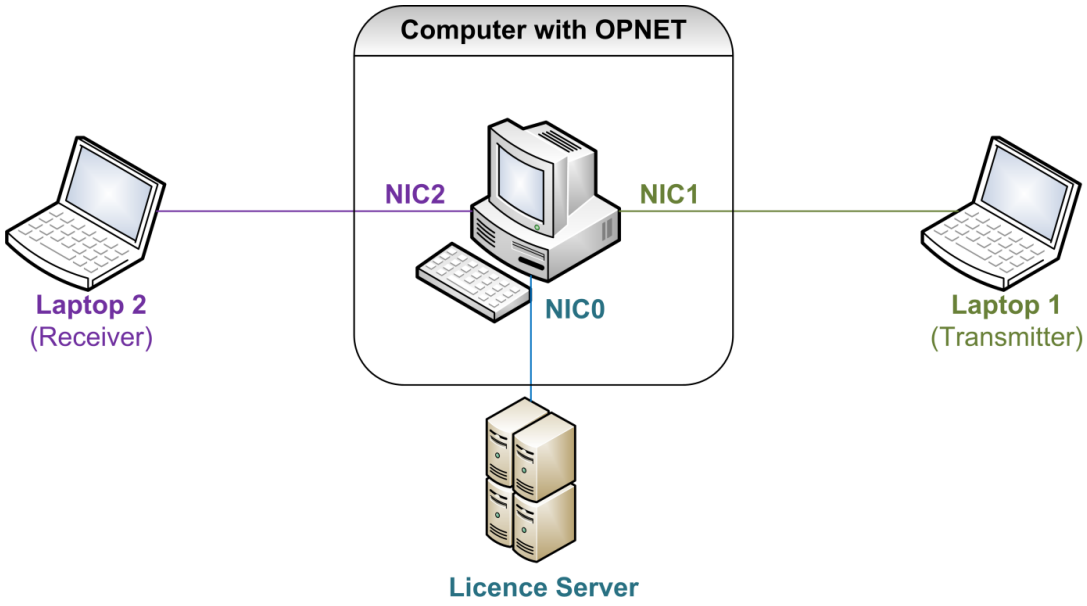


Figure 4.1: System-in-the-Loop Connections model

Figure 4.2: First scenario with no interference

The simulated network does not impose any restrictions on the communications channel other than those defined here. Two scenarios have been used. In the first scenario two Wireless LAN routers are connected using point to point connection with

a data rate of 1Mbps at the 2.4GHz frequency as seen in Figure 4.2. In the second scenario, 4 other nodes are introduced that operate in the same frequency as shown in Figure 4.3. The three workstations are streaming video from the server at the same data rate as that of the routers, i.e. 1Mbps. This interference increases the load that the routers have to handle and thus causes the point to point connection between the two routers to drop packets. The effect of this increased load (congestion) is observed on the packets being transmitted via HTTP.

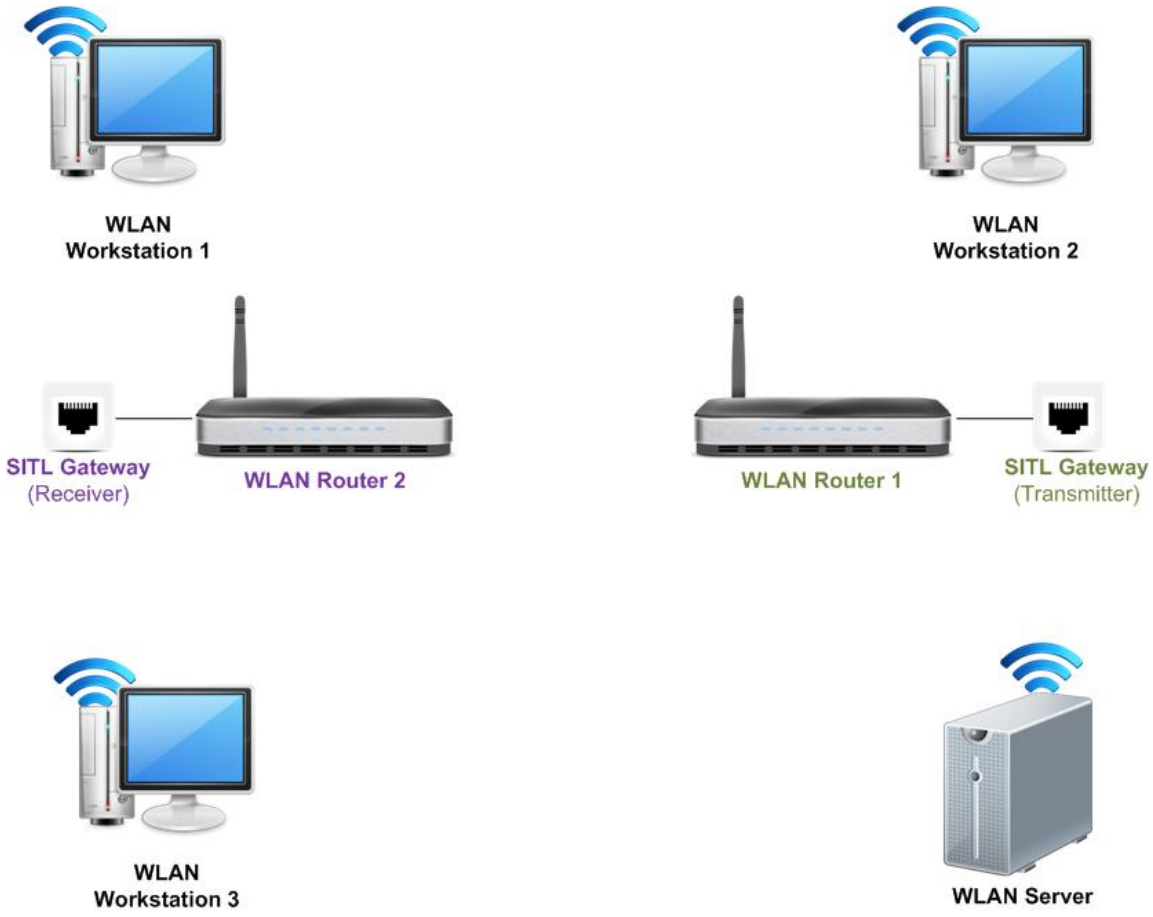


Figure 4.3: Second scenario with introduced interference nodes

4.4 Maximum Transmission Unit (MTU) Size

Every network has a maximum packet size that it can transmit, called the Maximum Transmission Unit (MTU). If the size of the packet is larger than the defined MTU, it has to be split into chunks that are equal to or less than the maximum size allowed before it can be transmitted. The probability of packet loss is increased when a large packet has to be fragmented because even if one of the fragments is lost, the whole packet might be rendered useless. MTU compliant packets also benefit from an optimized payload/header overhead relationship [95]. The maximum size of an IP packet allowed is 64 kilobytes [94] but it is rarely used outside research. In IP networks, the size of an MTU can range from 128 bytes to 10 kilobytes [95].

The size of the packet mutually agreed in a network based on the smallest MTU in the network is called Path MTU. But it is very difficult to determine an end-to-end MTU size between two IP nodes, and it may change dynamically between them during the connection [94]. In IP, being a packet switched network, each packet may take a different path with the need to determine a separate PMTU for each of the possible paths the packet could take. This is an additional overhead for a network. This has led to making the most common size used on the Internet to being 1500 bytes [94], because it is the size of Ethernet MTU. While not always the case, most end points in an IP network are connected to Ethernet; this becomes the largest size a network can connect.

Researchers have been trying to increase the size of the Path MTU by determining the lowest size a network would accept before fragmenting the received packets. These efforts led to the RFC on Packetisation Layer Path MTU Discovery (PLPMTUD) in

2007 [96]. In PLPMTUD, the maximum packet size on a particular path can be dynamically discovered through progressively transmitting larger packets.

4.5 Experimental Evaluation of Packet Size Variation

In the proposed solution, an MVC encoded bit stream is transmitted over HTTP as shown in Figure 4.4. The sequences are encoded using MVC. The encoded bit stream is passed to the HTTP packetiser where they are packaged in different sizes at the application layer before being passed to the physical layer for transmission. The process is applied to congestion-free network first and then repeated using a busy wireless channel. The bottleneck in the network is the wireless channel with a fixed bandwidth of 1Mbps. This bandwidth is competed for by the video transmitted as part of the simulation and the other nodes in the network which are set to communicate with each other in the set wireless domain.

The time of the complete video delivery is recorded for each packet size. Fifteen packet sizes are used for testing starting with 2^8 to 2^{22}. Although the most commonly used size of an MTU is 1500B, this data set is chosen to test the effect of both the smallest packet size of 256B to 4MB. The largest packet size IP allows is 64kB. This is much smaller than the packet sizes considered in the data set here. This is to study the effect of the increased sizes on the transmission rate when exceeded.

To establish a benchmark, the first view is transmitted through the network with no interference. The process is repeated three times. The average time is recorded for each video size. The system is further tested by the transmission of three views over the same network for the entire packet sizes repeated three times and the average noted. The final

test is the transmission of the first view through the network with interference from the other four nodes that are competing for the 1Mbps wireless bandwidth. The average of the three transmissions is recorded to compare with the benchmark.

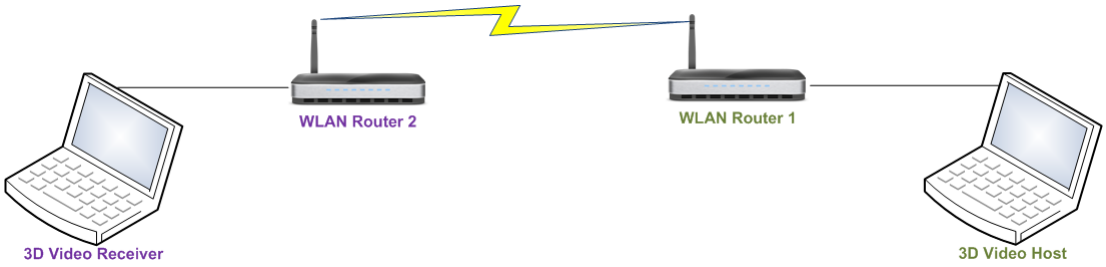


Figure 4.4: Proposed System testbed

Figure 4.5: Proposed System

To test the proposed system, an HTTP packetiser was developed, depicted by Figure 4.4 and Figure 4.5. The server advertises the videos available that can be pulled by the client node. The client sets the size of the packets when establishing a connection with the server and selects the bitstream it wants to download. The server sends the requested bitstream using the packet size defined by the client. The video used in the test is Exit using 3 views, each consisting of 250 frames. It is encoded with Quantization parameter set to 29 and Group of Pictures (GOP) to 12. The optimum packet size is discussed in the following sections. A packet size is considered optimum when it takes the least amount of time to deliver with the least number of redelivery attempts. Figure 4.4 shows the proposed system blocks.

4.6 Experimental Results

Figure 4.6 shows the delivery times variation between the different packet sizes. The blue and red lines indicate the results of the benchmark obtained from the interference free network for one and three views respectively. The green line indicates the transmission time of the test video when interference is introduced.

Figures 4.7 and 4.8 show the network load when there is no interference, and when the bandwidth of the wireless network is competed for by four other nodes respectively.

Figures 4.9 and 4.10 show the amount of data drop due to buffer overflow in the wireless network in the interference-free network respectively. Notice that there is no data drop in the interference-free network.

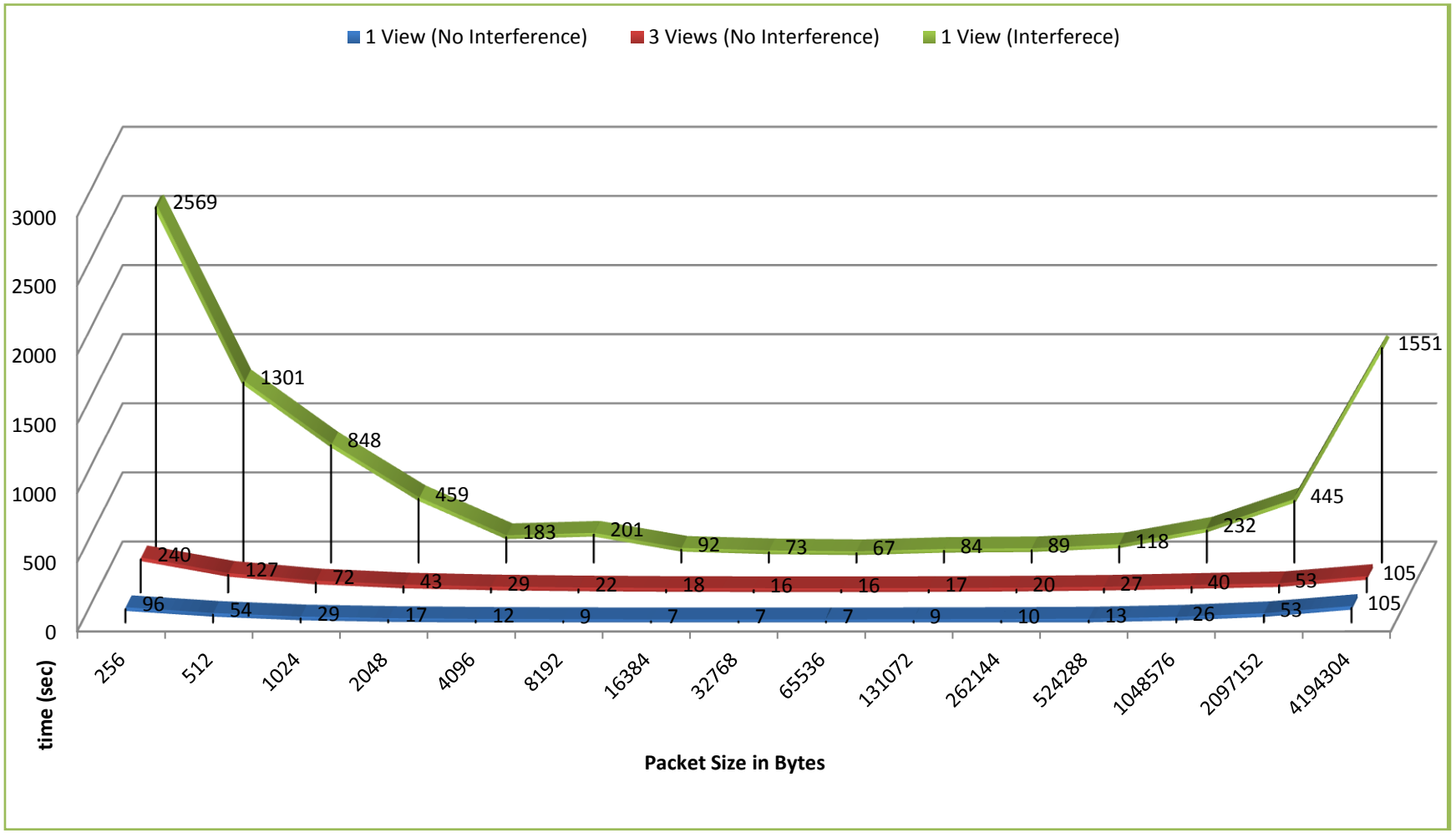


Figure 4.6: Delivery time variation depending on packet sizes for the network with interference in reference to the network with no interference (used as a benchmark)

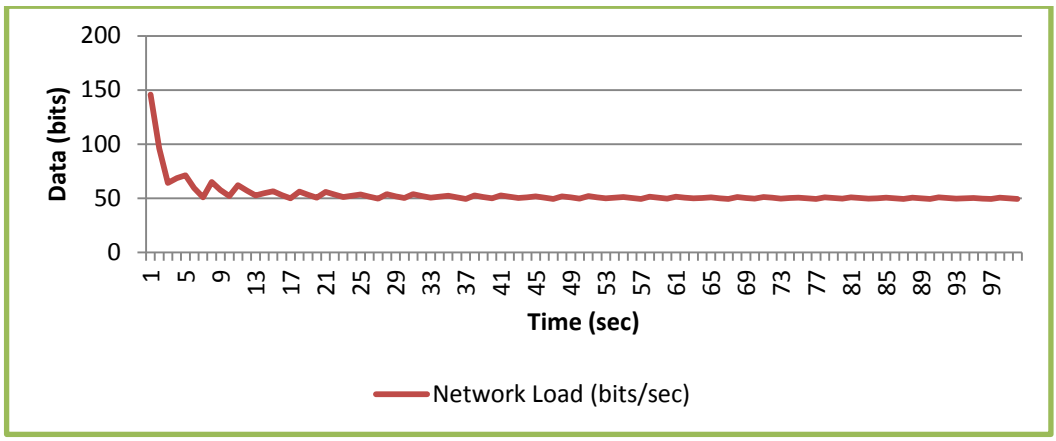


Figure 4.7: Network load when there is no interference from other nodes in the network

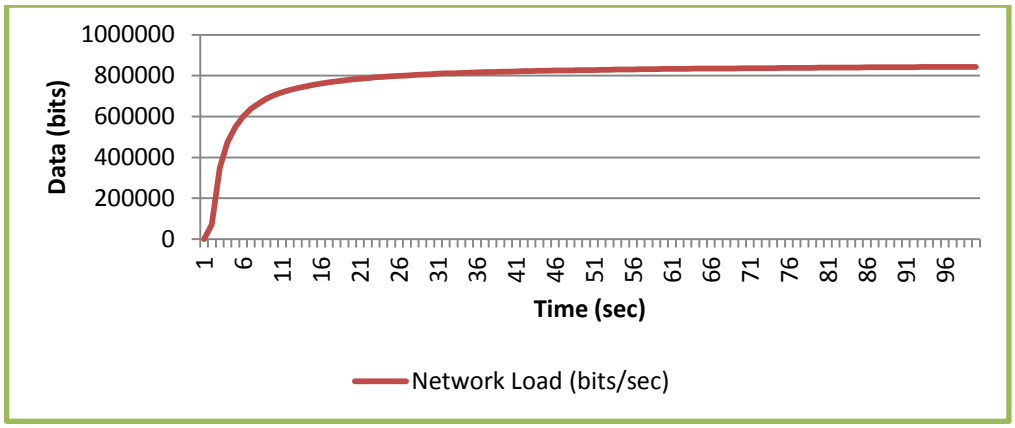


Figure 4.8: network load when the other four nodes in the network are competing for the wireless bandwidth in the network

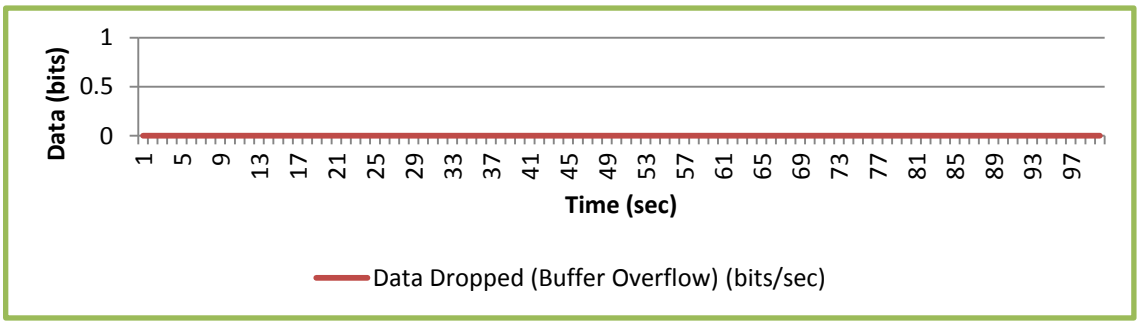


Figure 4.9: Data dropped due to the interference free network. Note that it is zero, meaning no data has been dropped

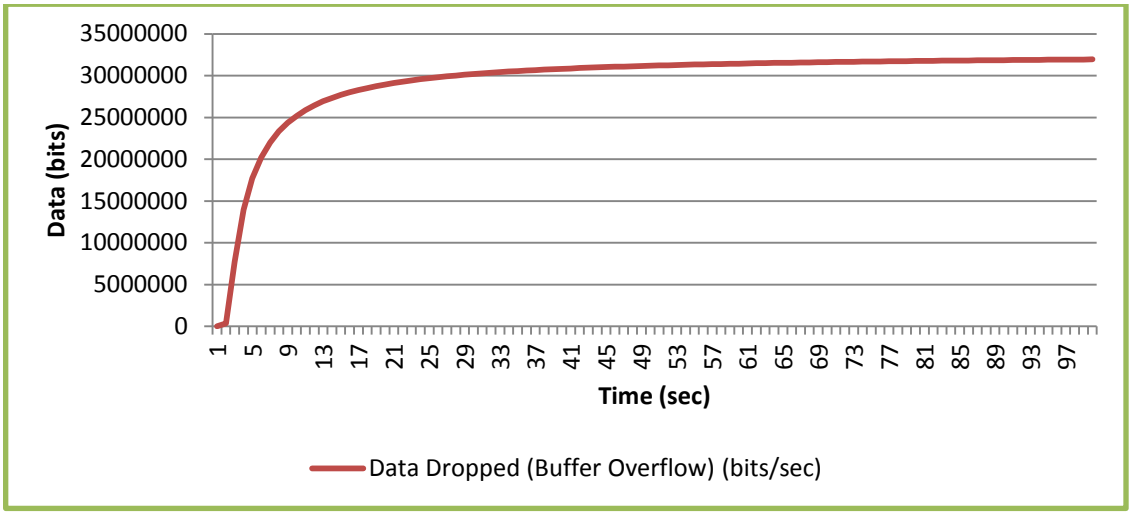


Figure 4.10: Data dropped in the network with interference from the other four nodes in the network

4.7 Analysis and Discussion

Starting with Figures 4.7 and 4.8, the data being transmitted on the network before any of the videos are being transmitted, paint a picture of the load on the network from other nodes. Figure 4.7 shows that after the initial handshake the network traffic is reduced to 55 bits/sec on average, which is the announcement and route discovery traffic. There is no other traffic going through the network. Figure 4.8 on the other hand shows the large amount of traffic going through the wireless network. This is the communication of the other four nodes on the network. Figures 4.9 and 4.10 enforce this picture with the data dropped by the wireless network due to buffer overflow. Note that there is no data drop when the other nodes are not transmitting but a huge spike in data drop can be seen when the other nodes start transmitting, indicating that the wireless network is already congested before any of the videos is transmitted as part of this research. This is the desired outcome that this research was looking to establish—a

congested network through which the delivery of MVC encoded video is transmitted over HTTP under different packet sizes.

In the test bed network, the most optimized results in terms of delivery time are seen when the packet size is 2^{16} (64 kB)—the size of maximum IP packet allowed [94]. This trend continues to the congested network. The amount of delivery time increases when the packet size is reduced or increased beyond 64kB. This increase is considerably more in the congested network.

Small packets take longer to deliver because of the increased overhead. They also perform badly in the congested network because there are more packets to be sent and hence more chance of loss. This increased packet loss adds to the spike in delivery times. There is also the matter of processing the packets in time. The network cards transmitting/receiving must have the capability to process more packets per second (PPS) as they arrive. The load on the processor or the sending and receiving node increases, not because there is extra load but because there are more packets to process.

Packets larger than the maximum size allowed for IP packets also increase the amount of processing time. Since the network card has to fragment all packets larger than 1500 bytes, because it is connected to an Ethernet network, the amount of data it has to hold in the buffer increases with larger packets and more processing needed. Although TCP can transmit a maximum of 64kB packets/segments, Ethernet (the network it is connected to) does not allow the transmission of packets larger than 1500 bytes. But the segmentation is faster at the physical layer than when handled by the application layer. The latter increases the levels of communication required for each packet delivery.

4.8 Conclusions

Transmission of the multi-view video requires more bandwidth compared to its spatially equivalent 2D counterpart. The reason for this is the increased number of views. H.264 MVC can encode videos with better efficiency by exploiting inter-view redundancy but it is still considerably more data to transmit.

HTTP is widely used for streaming video over the internet. Most video sharing sites, such as YouTube, Netflix, Vimeo use HTTP to serve their videos to clients. In a network where quality is an important factor, the video can be transmitted via HTTP which is built on top of TCP/IP at the cost of extra time. It guarantees the delivery and benefits from security clearances as it is not blocked by most firewalls. However, time is critical in the transmission. This chapter explored the idea of varying packet sizes to reduce the delivery time. Small packets are favoured by real-time video streaming and they are also used in mobile networks but they require more processing power and are more prone to loss in congested networks.

Large packet sizes take longer and if lost can affect the video but since in TCP it will be retransmitted, they are preferred. Moreover, the largest packet size that a network can handle is dictated by the Maximum Transmission Unit (MTU) of the weakest link in the network. MTU differs for each network. The experiments in this work used Ethernet with an MTU size of 1500 bytes.

Application designers should consider dynamic path discovery to understand the largest packet sizes allowed avoiding fragmentation of packets en route and the processing

power of the end-devices. Increased amount of packets will increase the required processing power.

This chapter focused on the transmission over HTTP which uses TCP and hence adopts a reliable transfer approach. All packets transmitted are guaranteed to arrive through repeat transmissions when packet erasures are reported. It is not an ideal protocol for time critical transmission such conversational video communication. But errors in the transmission of multi-view video propagate from one view to its dependent views. The next chapter explores the use of Real-time Transport protocol (RTP) and proposes a solution to minimise the effect of transmission errors on multi-view video quality.

Chapter 5: 3D Video Transmission over RTP

The use of inter-view prediction and hierarchical pictures in the Multi-view Video Coding (MVC) extension of the H.264 Advance Video Coding (AVC) standard produces better compression efficiency in multi-view video. This however, results in a bitstream that is considerably more prone to transmission errors than its simulcast coding counterpart. This chapter presents the idea of transmission of MVC coded multi-view video over multiple RTP sessions with experimental evaluation of its effectiveness. The results show that by separating the transmission of frames based on their level of importance and protecting the important streams, one can minimise the effect of transmission errors in the hierarchical B frames. This chapter also discusses the use of multi-session transmission in delivering a more adaptable bitstream where the number of streams can be adjusted based on the network conditions and, in extreme circumstances, all but the base view stream can be dropped to avoid further congesting the network and to deliver at least one high quality view to the end user. Our experimental results prove that in multi-session transmission confining errors to the RTP session encompassing the hierarchical B results in minimising the impact of errors on the sessions containing the base view and inter-view P frames; error rates as high as 10% have negligible effect on the quality of video received.

5.1 Introduction

Three-dimensional (3D) videos have been around in the entertainment industry for some time now with major blockbusters grossing more than their two-dimensional rivals. 3D videos are also becoming a common sight at home thanks to 3D sports channels, blue-ray discs and now the Internet. It is the Internet – with its inherent

flexibility, heterogeneous nature, and increasing bandwidth – that is expected to be the most widely utilised medium of delivery for 3D videos. This chapter explores the delivery of 3D videos encoded with the multiview video coding (MVC) extension of the H.264/AVC (Advanced Video Coding) standard using multi-session transmission (MST) feature of the Real-time Transport Protocol (RTP). Each RTP session is generated based on the number of priority levels required. A frame is important if it cannot be reproduced. Sessions with low priority level can be dropped based on network conditions. Depending on the network conditions and bandwidth availability, the number of streams can be adjusted, giving the MVC encoded stream the flexibility of simulcast without compromising encoder efficiency.

5.2 MVC NALU Interface

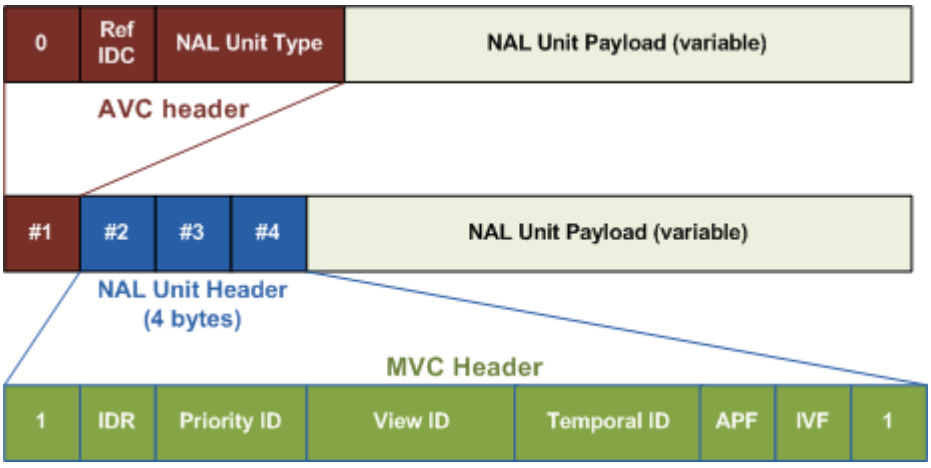


Figure 5.1: MVC NAL unit interface

MVC, standardised in 2009, covers a wide range of 3D video applications including 3D video streaming, free-viewpoint video as well as 3DTV [83]. It is an extension of the AVC standard. This makes it inherently backward compatible with AVC which serves

as the base-view that can be decoded independently in the absence of the MVC decoder [80]. Any additional views are referred to as enhancement views and are typically coded using inter-view prediction within the same bitstream [84]. Regardless of the view, the encoded video is packaged into the bitstream using Network Abstraction Layer (NAL) units which can be identified by its 1-B header. In its payload, NAL units carry the video coding layer (VCL) and non-VCL information which includes supplemental enhancement information (SEI), sequence parameter set (SPS), and picture parameter set (PPS). The 1 byte header is extended by 3 bytes in non-base view NAL units. The extended octets include, among other things, view ID and priority ID as shown in Figure 5.1.

This information is used to identify and build multiple streams with different priority levels as defined in section 3.3.3. One benefit of simulcast transmission is that it generates independently decodable bit streams [97] but it does not exploit inter-view redundancies (see Chapter 3 for more details). MVC generates more efficient bit streams but contains inter-view dependencies and the presence of all views in the bitstream even if the decoder can only extract AVC bitstream. Creating multiple streams based on priority level and different views at the NAL unit, an MVC bitstream can be transmitted with different levels of protection and make the extraction of the base view independent of the enhancement views. MST transmission gives the end-point the flexibility to drop enhancement views in congested and/or low bandwidth channel conditions. This will reduce the amount of traffic that needs to be transmitted in scenarios where the decoder can only decode the main view and where bandwidth restrictions do not allow for the transmission of multiview video. In the latter case, we

avoid further congestion of an already congested network with data that cannot be decoded without significant drop in the quality of the received video. In the first scenario, it makes the transmission more selective, e.g. the base view only is transmitted in the absence of an MVC decoder. This is further discussed in section 5.4.

5.3 RTP Payload for MVC

Real-time Transport Protocol (RTP) provides encapsulation of real-time media for immediate transport and consumption over the Internet Protocol (IP) using a variety of transport layer protocols such as the user datagram protocol (UDP), the transmission control protocol (TCP) or the datagram congestion protocol (DCCP) [84].

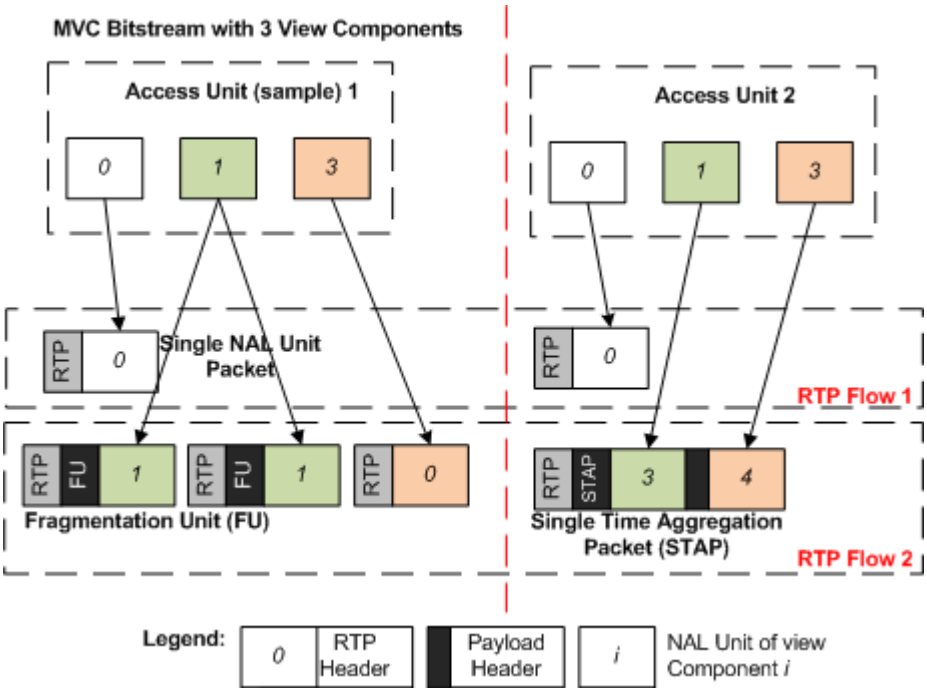


Figure 5.2: MVC multi-session transmission with two flows [84]

An Internet draft of the RTP payload is presented by the Audio/Video Transport Payloads Working Group which specifies how MVC NAL units can be encapsulated in

an RTP packet stream. It is built on AVC and some features of H.264/SVC (scalable video coding). An RTP packet can include a single NAL unit, a fragment of it, or multiple NAL units and in cases where the size of NAL units is smaller than the maximum transmission unit (MTU) size, multiple NALUs can be aggregated into a single packet. Where the latter is applied, the NAL units must belong to the same layer and access unit. Once the packets are formed, they can be transmitted in single-session transmission (SST) or multi-session transmission (MST) as depicted in Figure 5.2.

In SST, all views are transported with one RTP flow whereas in MST, multiple RTP flows deliver the NAL units from a single or multiple views. It is the latter that we employ in transmitting MVC bitstreams. In MST, each stream must have the same synchronisation source (SSRC) to associate the multiple streams with each other. Association can also be established using RTP control protocol (RTCP) Canonical Names CNAME [83]. The dependency between the RTP sessions can also be signalled via session description protocol (SDP).

5.4 Video Transmission

Multiview video as the name suggests has two or more views, which increases the encoding complexity and bandwidth requirement for transmission. There are two techniques that can be used to transmit these views. The most basic is to treat each view as separate and encode them separately as depicted in Figure 5.3. This technique is called simulcast as each view is independently encoded.

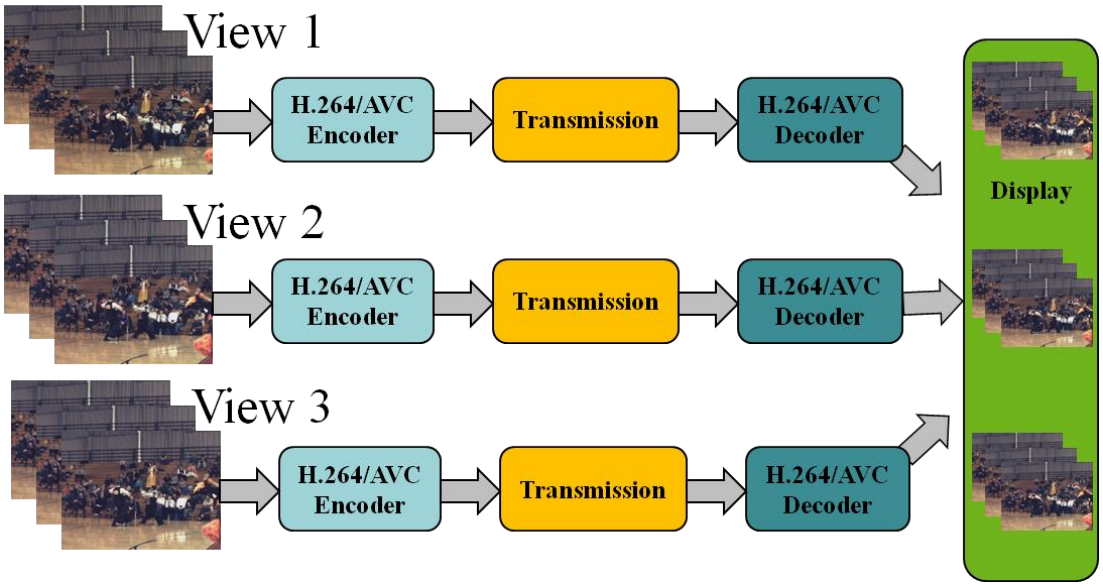


Figure 5.3: Simulcast transmission with three views

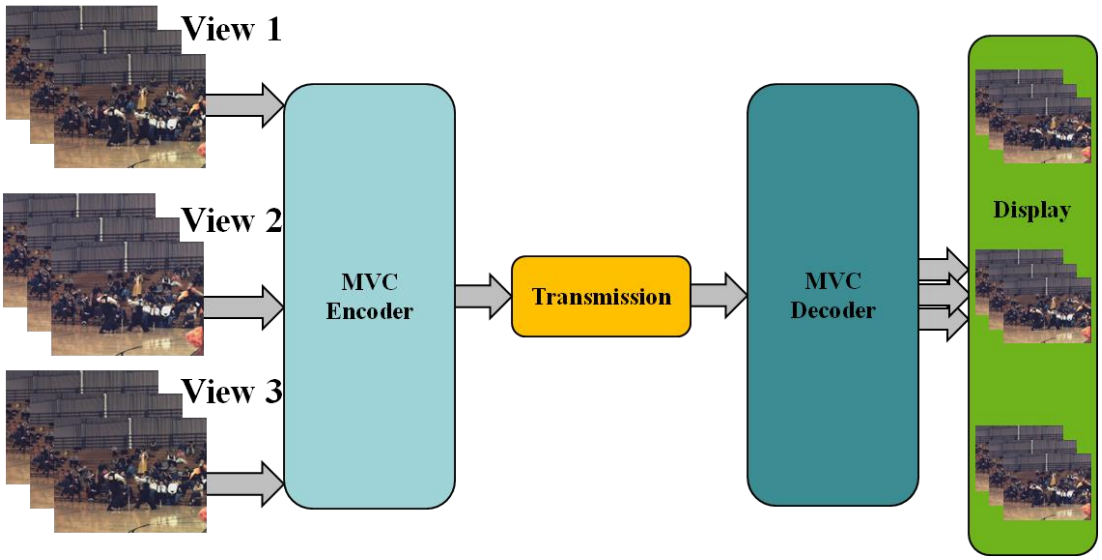


Figure 5.4: Multiview transmission with three views

Figure 5.5 shows prediction when each view is encoded independently. While this method has its benefits, it does not exploit the redundancies that are preset in between the views. Figure 5.4 shows the multi-view encoding where three views are

incorporated for better encoding. While it exploits the similarities that are present between the three views, it increases the effect of transmission errors. Figure 5.6 shows the inter-view prediction when the three views are incorporated. The inter-view prediction produces a more efficient bitstream but one that is more vulnerable to transmission errors.

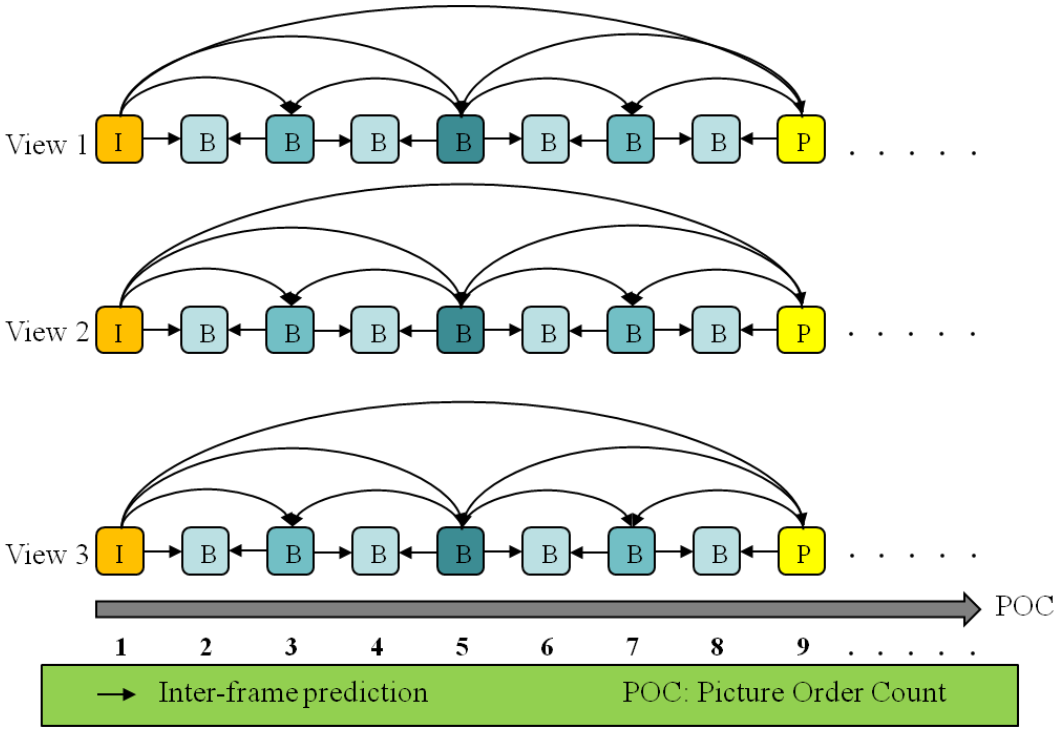


Figure 5.5: Prediction in each view when encoded independently

Since MVC is an extension of H.264/AVC and needs to have at least one view which is backward compatible with AVC, it uses the AVC encoder to encode each view and then assemble them in the time-first coding [99]. The interdependency between the views makes switching views difficult as the frames that the required view depends on for inter-view motion prediction will also need to be transmitted. The time first coding structure allows the decoder to decode pictures belonging to different views in the same

time domain. This, however, introduces issues when transmitting with multi-sessions as the pictures need to be put back in the right order for the decoder to be able to decode and display the videos in the correct order, as shown in Figure 5.7. Although it exposes the bitstream to transmission errors, the time-first coding of frames is used as a solution to the multisession transmission synchronisation. Frames received from the different RTP sessions are synchronised in the order of display before being fed to the decoder.

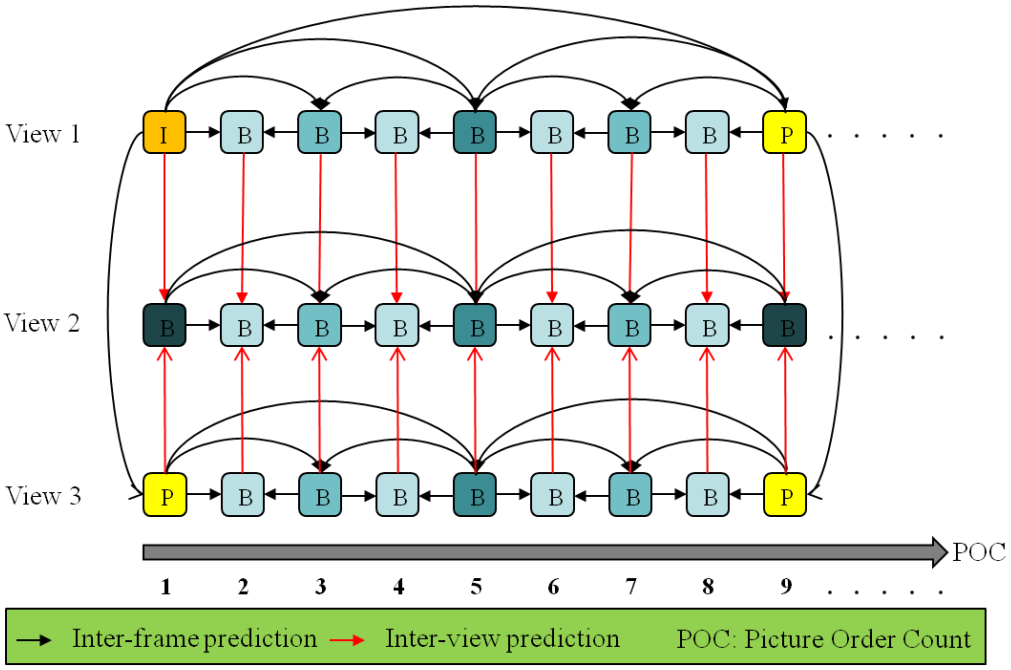


Figure 5.6: Prediction structure when the three views are encoded using inter-view prediction in addition to temporal prediction in each view

Video transmission of MVC encoded bitstream has been an active research area. Several publications have studied the effect of errors and error concealment techniques. However, these studies have focused on single channel transmission. To the author's knowledge, MST of MVC encoded video has not yet been experimentally explored for 3D video transport in available literature. Interested readers can refer to [94] for

streaming H.264/AVC video over IP networks. The effect of packet loss on the quality of streamed MVC videos is studied in [102]. Error concealment techniques for MVC encoded video sequences are evaluated in [103].

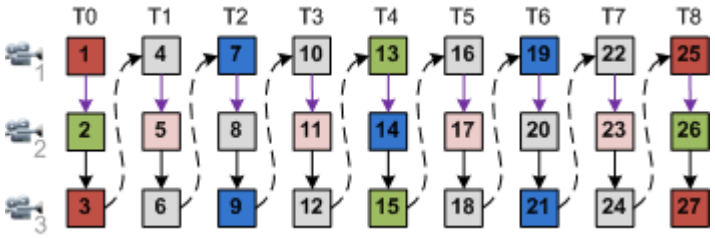


Figure 5.7: Time-first coding

5.5 Multi-Session Transmission in RTP

In order to see the effect of transmission errors in single view and the proposed multi-session transmission, a model of the transmitter and receiver nodes were produced using the open source multimedia streaming application, Sirannon [100]. Sirannon is a testing tool accepted by the Video Quality Expert Group (VQEG) for streaming video sequences and simulating network impairments. Three videos of the MERL sequences, Ballroom, Exit, and Vassar, were used for testing and encoded in accordance with [102]. The numbers of views used were three, each containing 250 frames, totalling 750 frames per multi-view video. The views were encoded with base QP of 31, 29, and 30 respectively. The sequences were tested for Group of Pictures 4, 8 and 12 to make the tests more comprehensive.

First, the video was transmitted in a single view session with packet loss 0%, 0.5%, 1%, 2%, 3%, 4%, 5%, 7%, 9%, and 10%. The video was then split into three RTP sessions. The first view, which is independently decodable, was transmitted in a separate session,

the P frames in view 2 were sent in another RTP session and all of the hierarchical B frames from views 1 and 2 were sent in a third RTP session. For each packet loss test, 10 transmissions were made. A total of 1800 tests were done, 900 using each transmission method, i.e. single and multi-session. The best of the 10 tests for each transmission were selected for comparison.

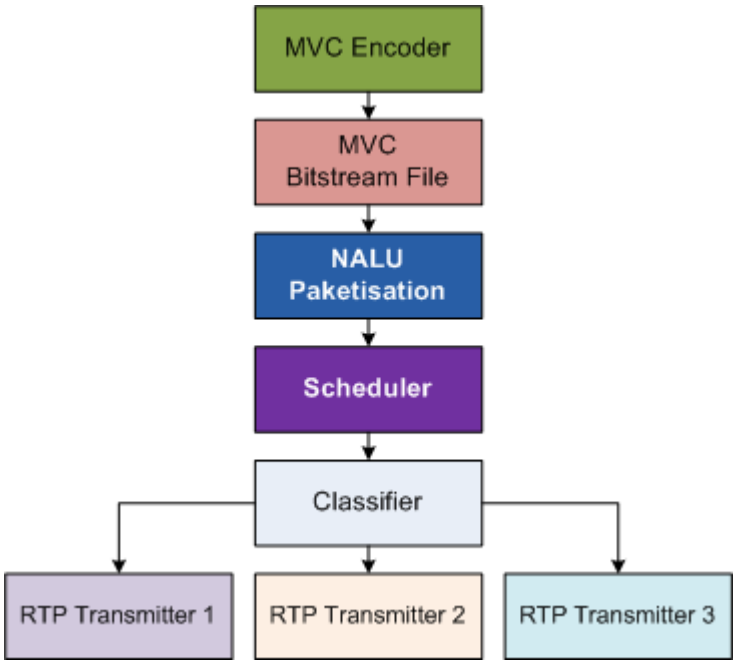


Figure 5.8: MST transmission test bed

The experimental test bed for multi-session RTP transmission is depicted in Figure 5.8 and Figure 5.9.

In the multi-session transmission, only the RTP session with the hierarchical B frames session was subject to error loss. A frame classifier was used to identify each frame before they are forwarded to the relevant RTP transmitter. A frame scheduler was used

to synchronise the two receiving RTP channels into a single stream ready for decoding. The maximum MTU size was set to 1500 Bytes. NAL units larger than the maximum MTU size were fragmented as described in section 4.4 and in accordance with RFC 3984 (RTP payload format for H.264 video).

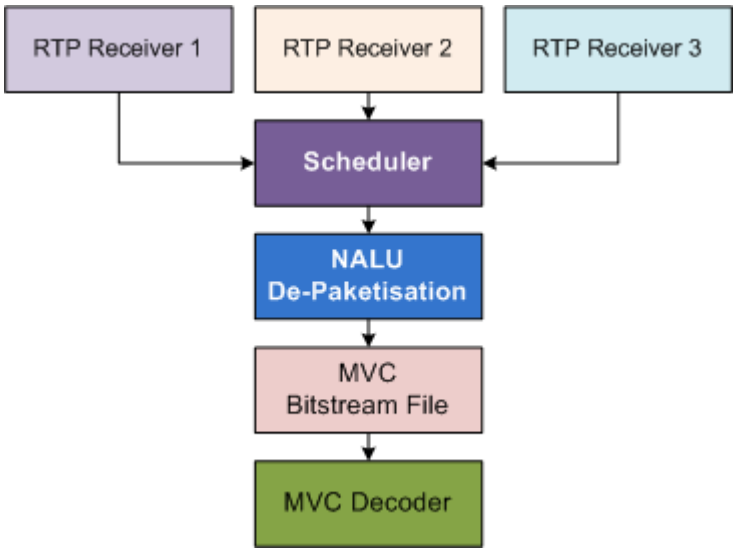


Figure 5.9: MST receiver test bed

JMVC reference software version 8.5 was used for the encoding and decoding of the multi-view videos. Since H.264/AVC decoder only accepts complying bitstreams, the decoder was modified to accept corrupt bitstream by using frame copy to replace the missing frame. This is done by inserting the previously decoded frame if and when available in the decoder buffer.

The quality of the video is assessed with the objective quality assessment widely used in 2D videos, PSNR. Since we are interested in the quality of each view, PSNR is a suitable measurement to evaluate the effect of errors in a particular view.

5.6 Experimental Results

Despite changing the code in the JMVC decoder to accept erroneous bitstreams, the decoder failed to decode some bitstreams in two conditions: (a) where the loss in transmission was in a frame that was either a reference frame; or (b) when the decoder buffer over loaded. The latter was more frequent in large GOPs.

Decoder failure occurred when a reference frame was dropped and the decoder lost synchronisation, making it unable to continue decoding the bitstream. In the multi-session transmission, the base view is kept error-free whereas the hierarchical B frames in the second and third views are subjected to packet loss. Due to the fact that B frames are bi-predicted, they were re-generated in both of the enhancement views despite the high packet loss.

As evident from Figure 5.10 compared to Figure 5.11 the number of decodable frames increases with multi-session transmission by threefold, especially at high error rates. Videos in MST scenarios were decoded in their entirety when the scenarios were GOP 4 and GOP 8. The few missing frames from the total 750 are those that were lost, but it can be seen that the decoder succeeded in continuing to skip past the errors. This was not the case in the 12 GOP scenarios at high error rates. The reason for this was the requirement for keeping a larger number of reference frames in the memory causing the decoder to crash in the case where there were a large number of errors in the hierarchical bi-predicted frames.

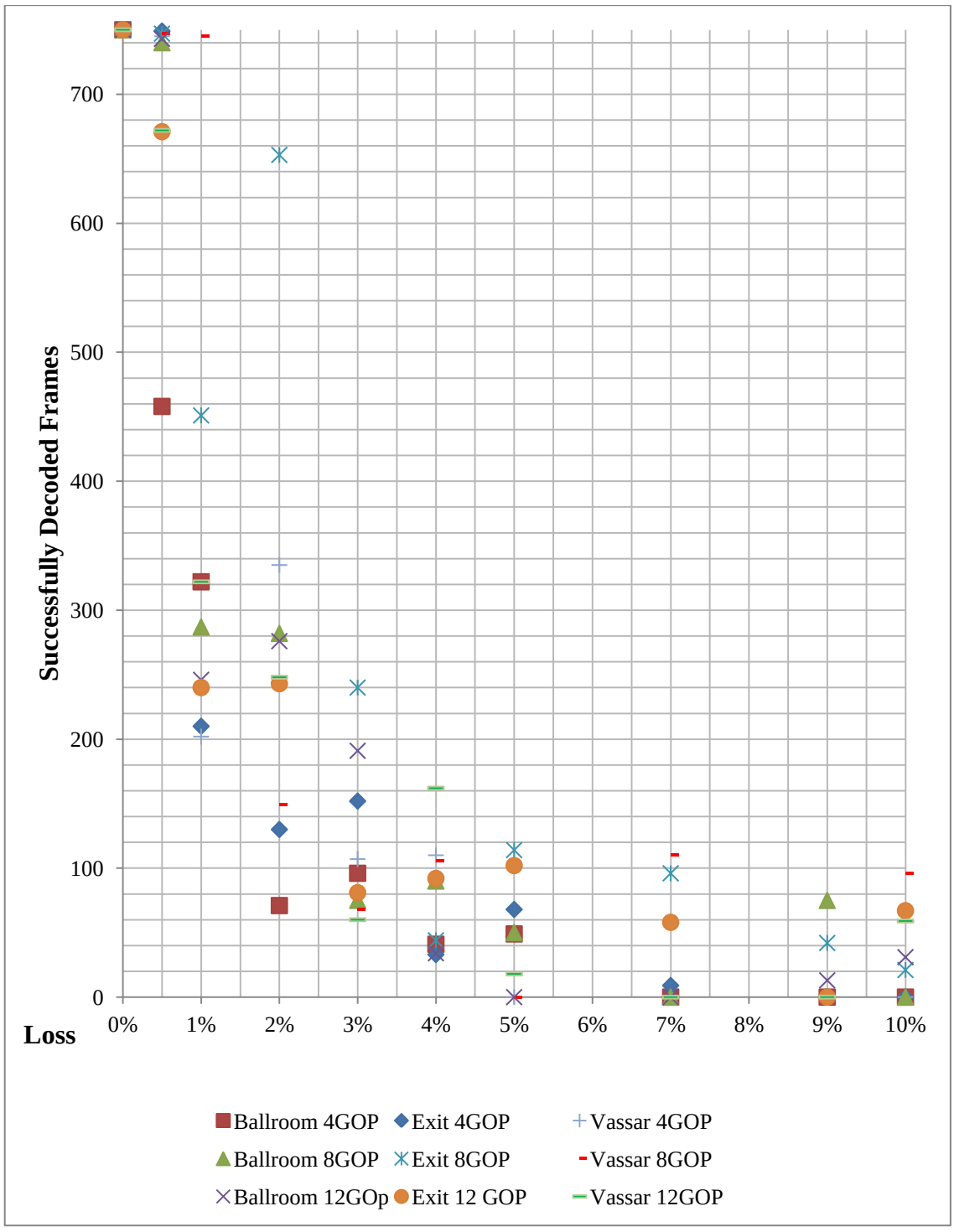


Figure 5.10: Graph shows the number of frames decoded for each error loss pattern in single session transmission

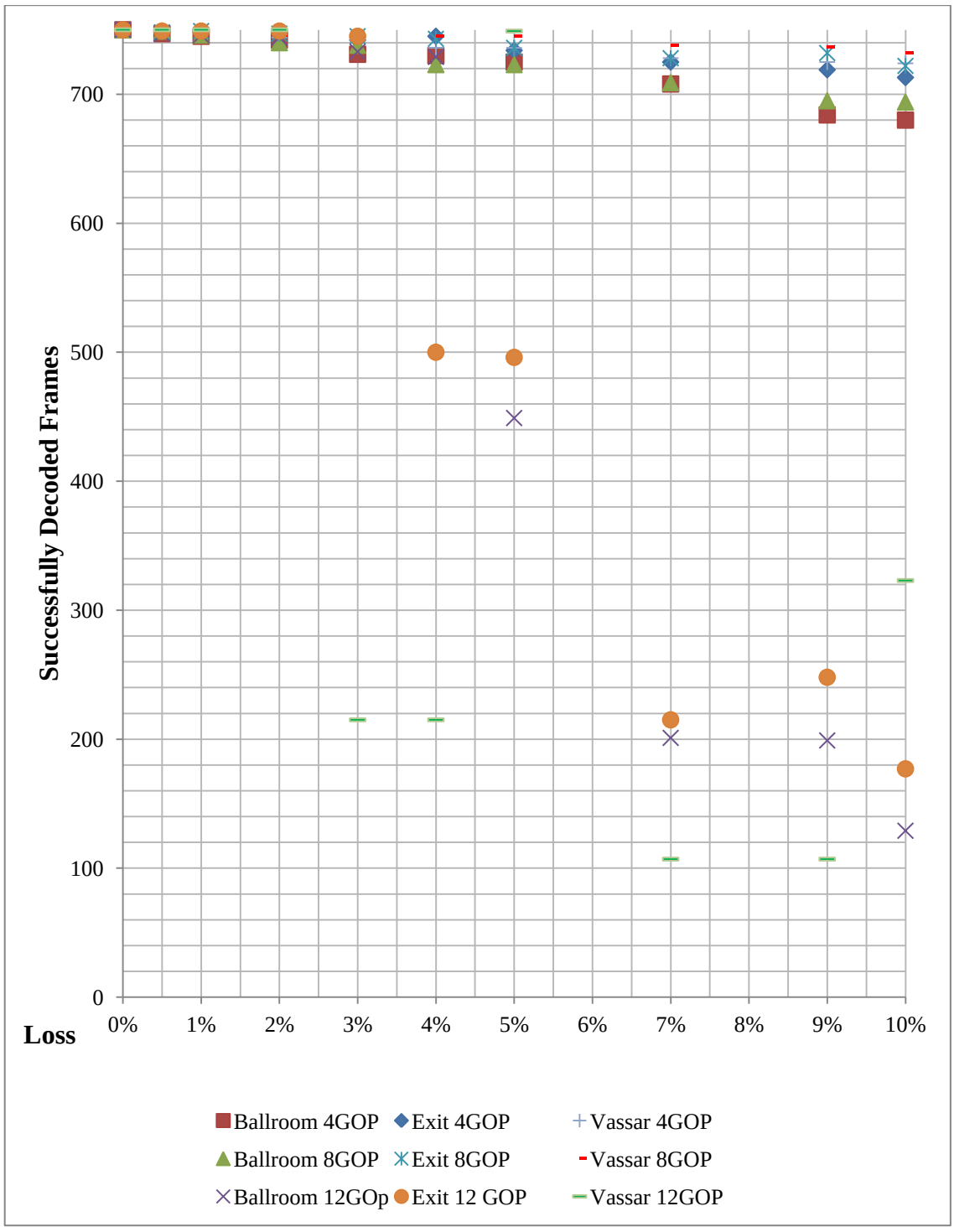


Figure 5.11: Graph shows the number of frames decoded for each error loss pattern in multi session transmission

PSNR results shown in Figures 5.12 and 5.14 further strengthen the case for using separate channels for different frames in MVC transmission. The optimum PSNR achievable for the MVC streams is demonstrated by the zero loss bitstreams. MST streams show a steady decline with the increase in the error rate. Since, in MVC, each view contributes to the quality of experience, the average of the three views is considered in the graphs shown in Figures 5.12 to 5.14. While in MST, the average PSNR drops below 25 dB—which in Mean Opinion Score (MOS) translates to an annoying level of impairment—in the high moving video, Ballroom, the other two videos stay 30 dB for most of the tests. In SST this decline is more rapid, despite the fact that only those frames that are decodable were compared. In instances where PSNR is zero reflects the fact that none of the 10 transmission tests for that video contained one which was decodable.

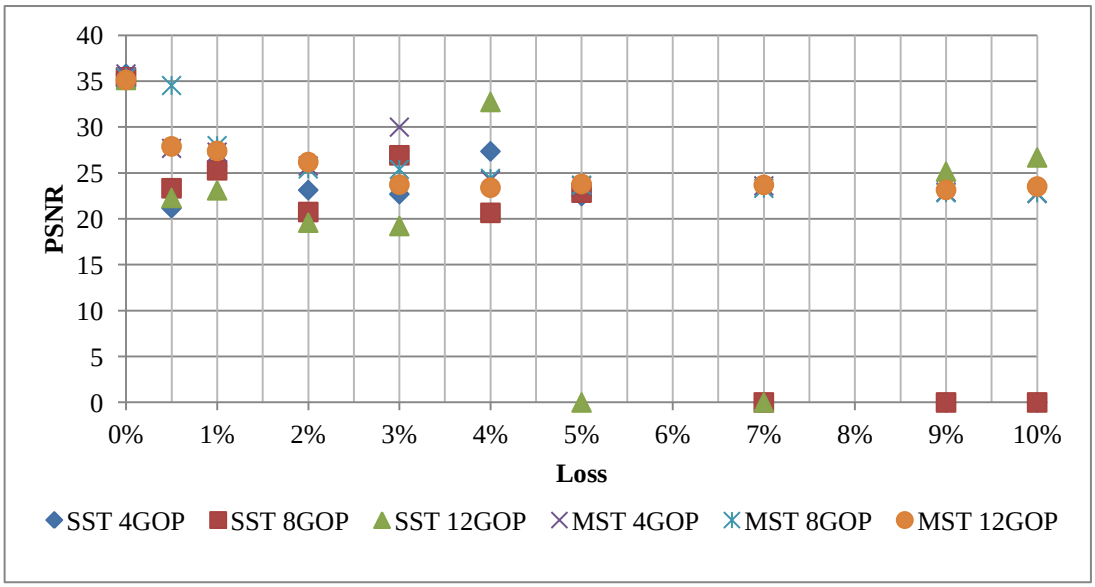


Figure 5.12: Average PSNR for each error loss pattern in single session and multi-session transmission for Ballroom sequence

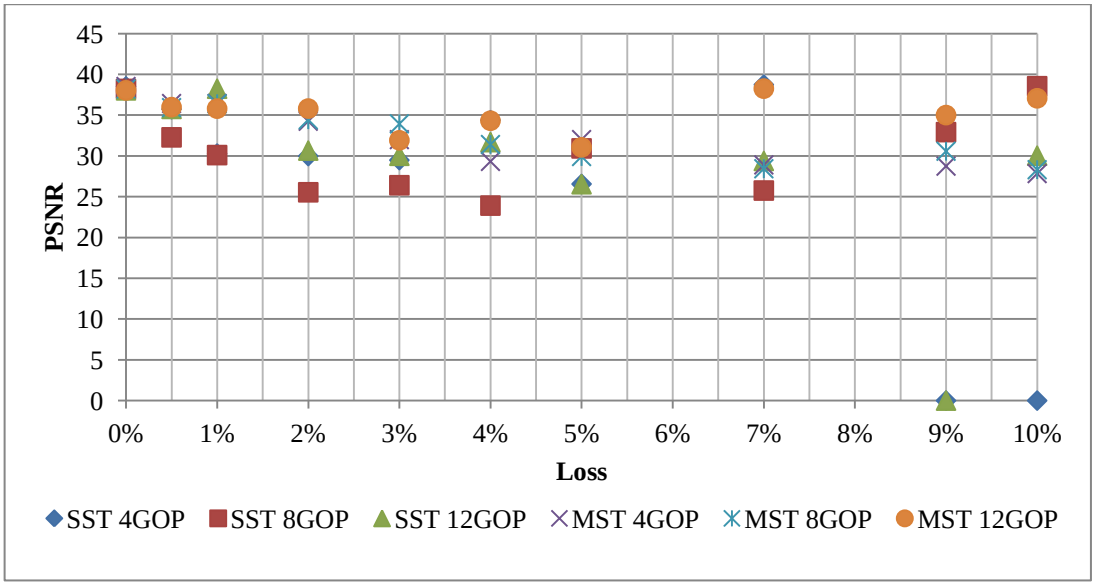


Figure 5.13: Average PSNR for each error loss pattern in single session and multi-session transmission for Exit sequence

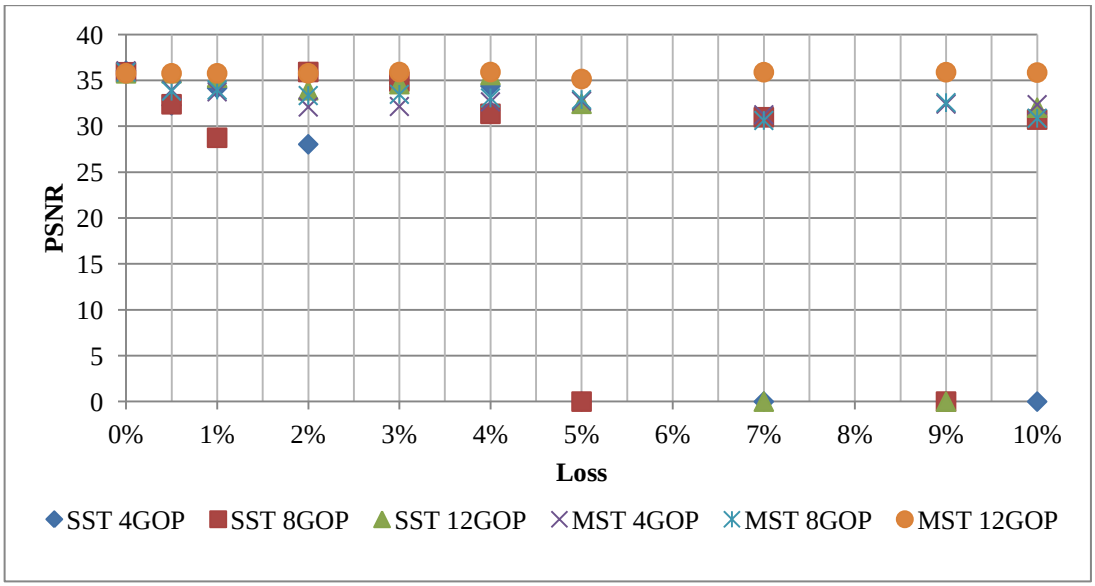


Figure 5.14: Average PSNR for each error loss pattern in single session and multi-session transmission for Vassar sequence

It must be noted that the quality degradation is consistent in SST and MST at comparable rates until the packet loss increases beyond 3%. The best performing video in SST is in the highest (12) GOP scenario but even then it is not predictable.

The effect on the quality of video is different at every simulation run. This is due to the random errors that are introduced affecting the bitstream differently each time. The effect is also different given the type of video tested. For example, in Vassar which has low object motion, the effect of transmission loss is lower. This is because the frame copy that was employed to recover dropped frames has a higher likeness to the dropped frame than it has in the medium and high motion sequences of Ballroom and Exit respectively. In the multi-session transmission, the quality is, in most cases, comparable to single session at 1% loss and slightly lower at 3% loss. But the major difference is its ability to resist errors far beyond 3%. Even at 10%, the PSNR value stays above 30 dB, except in the high motion video sequence—ballroom.

The reason the second and third views are able to stand errors in the multi-session transmission at considerably higher loss percentage in the hierarchical B frames, despite there being far more hierarchical B frames than I and P frames, is because they can be recreated. Only B frames depend on other B frames for prediction. I frames can be independently decoded and P frames only depend on the preceding I frames. This experimental evaluation proves that multi-session transmission can improve the quality of the bitstream when the base view and key frames are protected by separate transmission streams.

5.7 Analysis and Discussion

Multi-view videos have a large number of data with inter-dependent coding. Errors in a frame in one view can propagate to other views. The effect of these errors can be much larger when the following frames are bi-predicted from the frames where the errors occurred. Multi-view videos, as shown in this chapter, heavily use bi-predicted frames. This makes them more susceptible to error propagation resulting from errors in reference frames. Protecting the reference frames can improve the quality of the complete multi-view video content.

Decoupling the reference frames from the bi-predicted frames for transmission over separate sessions presents the problem of out-of-order-arrival of the frames. MVC as discussed employs time-first coding, which means it needs the frames in time order to decode them. Out-of order arrival of frames in a multi-view video during multi-session transmission is a certainty. Frames will arrive before those that are needed before it. RTP is therefore the right protocol to use, as it contains within itself certain mechanisms such as timestamps and control mechanism which can be used to establish the synchronisation needed at the end-point in such arrangement the frames are put in the correct order for decoding. Conventional transport protocols such as TCP, which is used in Chapter 4, are not designed for real-time transmission.

The content of a multi-session transmission can vary depending on how the video is segmented. In this work, the segmentation is based on the importance of data in multi-view video with respect to adaptable transmission channels. The 3-view multi-view video is segmented into three distinct sessions.

The first session being the base view which can be independently coded, should the end-device be a 2D video decoder or when the network is congested. The end-device can drop the other views without incurring any quality loss on the base view or further congesting the network with data it cannot decode in time for display. This makes every multi-view video stream adaptable. In addition, the bitstream extraction is already conducted so the decoder does not need to perform further bitstream extraction to display the base view.

The second and third streams contain all the predicted and bi-predicted frames for the auxiliary views respectively. This stream separation is to protect the more important predicted frames from the less important bi-predicted frames. Errors in the predicted frames can propagate exponentially in the bi-predicted frames but the same is not true vice versa. Hence, the stream containing the predicted frames can be protected to reduce errors.

The experimental results show that multi-view video decoders can handle errors in the bi-predicted frames and tolerate a high error loss. The results presented above show that almost all of the frames, excluding the ones dropped as a result of network congestion, are decodable even at a 10% loss rate. The effect of errors in bi-predicted frames, however, increases with the increase in GOP size. There are a number of factors for this, the most important of which is the decoder buffer. With a high GOP the decoder must keep more frames in memory for a longer period of time to serve the decoder needs for prediction. This leads to buffer overflow when the error rates increase. In this work, in case of an error in a frame, the previous reference frame is used to replace it, but it may already be flushed out of the buffer before the error occurs, which leads to

the decoder failure. This problem, however, is not critical and can be fixed by using error control techniques such as Forward Error Correction (FEC). The increased error rates also highlight the importance of selecting the right GOP size for a stream. Coding efficiency and transmission errors must both be considered when selecting the GOP structure during the encoding process.

Multi-session transmission of multi-view video can render the video stream more adaptable to the end-devices and offer additional error resilience capabilities by confining the errors to a particular spatial region. Experimental results shown above prove MST's effectiveness in handling error rates as high as 10% with negligible frame loss when the errors in transmission are contained to the relatively less important bi-predicted frames.

5.8 Conclusion

MVC greatly reduces the amount of data that needs to be transmitted by exploiting inter-view redundancies between adjacent video sequences in addition to the regular temporal and spatial redundancies in each view. This, however, makes it more vulnerable to transmission errors. Errors in key frames can render the entire bitstream un-decodable.

The inter-view dependency also means that errors propagate to other views when errors occur in frames that are used by other views for motion compensation prediction. The transmission of MVC bitstreams over multiple streams based on priority and view changes improve the quality of 3D video.

MST gives the end users/systems the flexibility to drop views in congested networks in order to allocate more bandwidth to the base view for better QoE. MST can also be used to reap the advantages of simulcast without losing the benefits of inter-view similarities exploitation.

It does, however, introduce problems with the synchronisation in erroneous channel conditions when packets arrive out of bound. This problem has been resolved by using the frame order to resynchronise by taking advantage of the MVC time-first frame ordering.

The focus of Chapter 4 and 5 was on the transmission of multi-view videos as produced by the current encoders. The next chapter tries to evaluate the use of virtual view rendering techniques in MVC, hence reducing the number of cameras required for

video capture and further increasing the coding efficiency while reducing the bandwidth requirements.

Chapter 6: Decoder Optimisation in MVC

Multi-view video delivery, as discussed in previous chapters, requires the transmission of a large number of views. H.264/MVC does a good job in compressing the views by exploiting inter-view redundancies. This, however, still puts the bulk of the load on the encoder/transmitter with little effort on the part of the receiver. With the increase in the processing power of end points, be they mobile phones, PCs, or even smart TVs, it is imperative to take advantage of this processing power. This chapter focuses on solutions to eliminate the need for transmitting all views in an MVC sequence, by recreating the views at the receiver using the full resolution adjacent decoded views to recreate the middle view without the need for any auxiliary information to be delivered. This chapter evaluates the complexity of different virtual rendering solutions and compares the quality of their results against each other and against the actual decoded view.

6.1 Introduction

Multi-view videos are recorded by cameras very close to each other, capturing the scene from slightly different angles. The video sequences used in most MVC research studies are the MERL sequences, Ballroom, Exit, and Vassar [104]. This research also employs these sequences. They are captured in 640x480 Bayer with a 19.5 cm baseline between optical centres, with the optical axis being parallel to the ground plane at approximately 1.5m alleviation from the ground up [104]. They are hardware synchronised so the time each camera captures a frame is the same. Figure 6.1 shows the camera setup used by MERL to capture their sequences.

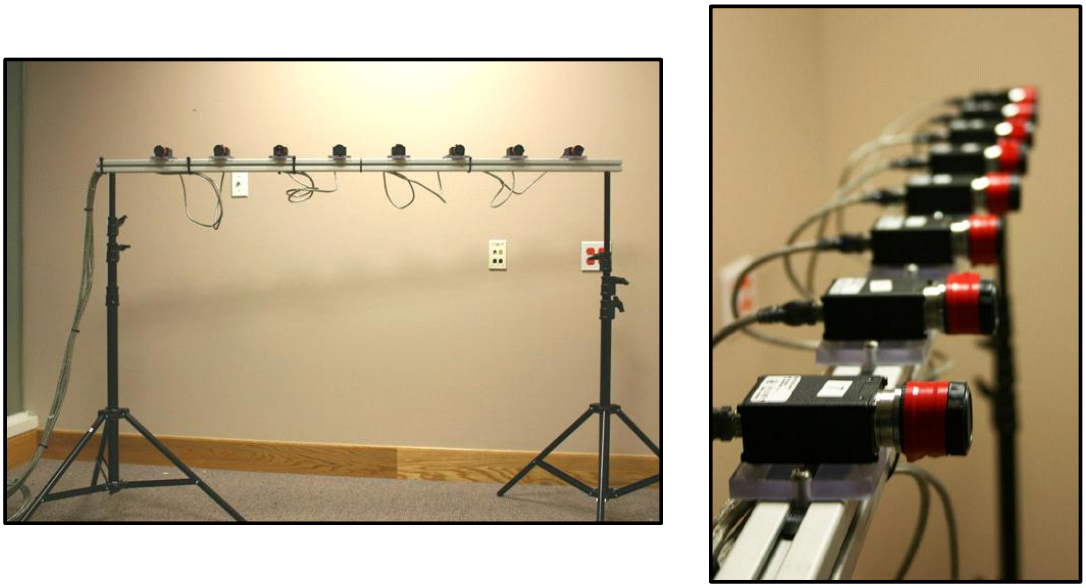


Figure 6.1: Illustration of 8 camera setup in Multiview Video Test Sequences from MERL [83]

As a consequence the inter-view similarities can be exploited to reduce the number of cameras needed for capturing a scene. This will greatly reduce the amount of traffic that needs to be transmitted across networks, thus allowing better utilisation of the available bandwidth. For example, if for every three views, one of the views can be set to be a virtual view that is recreated by the decoder. This means that the same amount of traffic that three views would normally generate can transmit five views, as depicted by Figure 6.2. Capitalising on the compression capabilities of MVC, this would reduce the load on the encoder at the cost of extra processing required by the decoder—effectively balancing the load between the server and the client.

Figure 6.2 illustrates the compression of five sequences using the H.264/MVC encoder, with the view 2 and 4 excluded from the process, making view 3 as the middle view—

which would otherwise have been view 2. Virtual view 2 and 4 will be recreated by the decoder from views 1 and 3, and views 3 and 5 respectively.

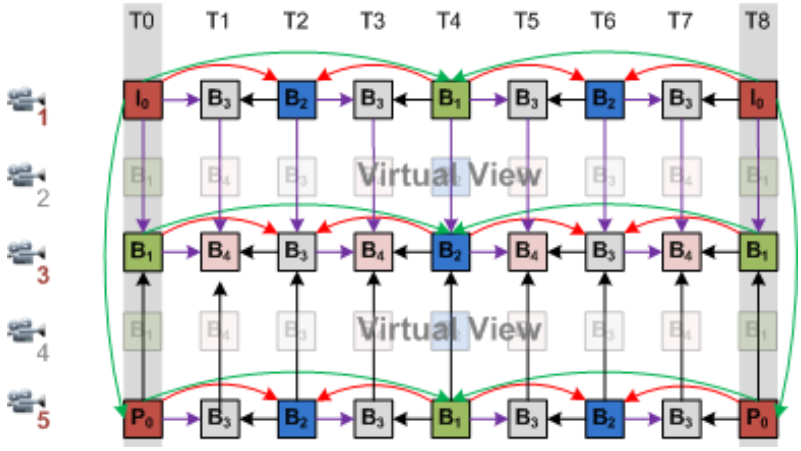


Figure 6.2: Depicting virtual views to further improve the compression ratio of the H.264/MVC

In order to reconstruct the virtual view by the receiver, the frames of the two reference views must be decoded. Since MVC employs a time first approach when decoding views, as highlighted in Chapter 3, the virtual view creation module can simply be inserted after the decoding of the frames in each time block. The process needs to be simple so that it does not add too much extra delay and does not considerably increase the processing power required.

6.2 Extracting Virtual View

Similarities between the frames of the views are visually demonstrated by Figure 6.3, where Figure 6.3 (a) represents view 1, Figure 6.3 (b) represents view 2, and Figure 6.3 (c) represents view 3. As can be seen the details required in view 2 are present in view

1 and 3. The process of generating the views is described in this section. Three methods are devised: Stitching, Disparity Prediction, and Panoramic Imaging.


(a)


(b)


(c)

Figure 6.3: Frame one of Vassar Sequence demonstrates the similarities between the frames (a) View 1, (b) View 2, (c) view 3

6.2.1 Stitching Virtual View

The first method proposed is simple with low algorithmic complexity--remove the excess pixels from the frame of view 1 and view 3 and then stitch them together to form the virtual view.

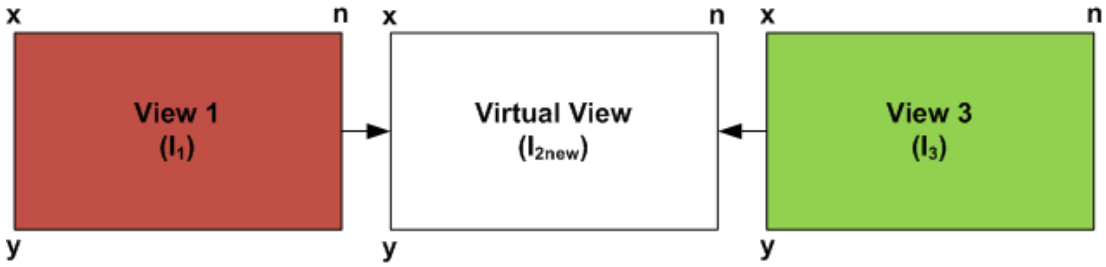


Figure 6.4: Creating virtual view from one adjacent view on each side

Assume I_1 and I_3 represents the frames of view 1 and 3 respectively and I_v represents the virtual view. ω is the size of the slice containing the excess pixels. To create the virtual view, we will first need to remove the excess pixels ω from I_1. The rest of the image, $\omega \; to \; n$ are shifted to position x_0 to form the first part of the virtual view. Equation 1 represents this process mathematically and Figure 6.5 graphically.

$$I_V(x, y) = I_1(x + \omega{:}\, n, y) \qquad \text{Equation 1.}$$

Where n = size of image along x-axis, and ω is a variable length difference between consecutive views along x-axis. ω is calculated through trial and error for each video once, at the start of the process.

Figure 6.5: Extracting the relevant parts of view 1 to make the first part of the vitrual view

The pixels in the I_3 to complete I_v are between $n-2\omega$ to $n-\omega$. These pixels are appended to I_v to complete the virtual view frame. Equation 2 represents this process mathematically and Figure 6.6 graphically.

$$I_V(x+\omega-n,y) = I_3(x+n-(2\omega):n-\omega,y) \qquad \text{Equation 2.}$$

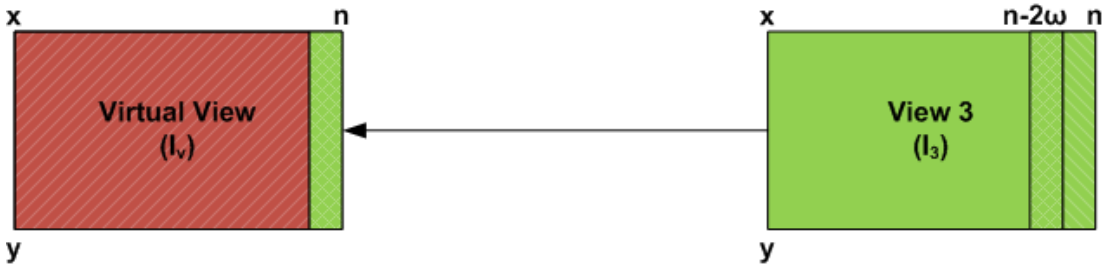


Figure 6.6: Extracting the relevant parts of view 3 to make the remaining part of the virtual view

It must be noted that the above steps must be applied to each colour channel separately, i.e. the process to be repeated for R, G, and B channel individually. Applying the above process to frame 1 of the Vassar video sequence, we get the result shown in Figure 6.7. To an untrained naked eye this frame looks very similar to the original frame but

objective test using PSNR produces 22.5 dB. Its MOS is below the very annoying threshold of 25dB, where the same frame has a PSNR of 36.6 after being encoded/decoded using MVC.

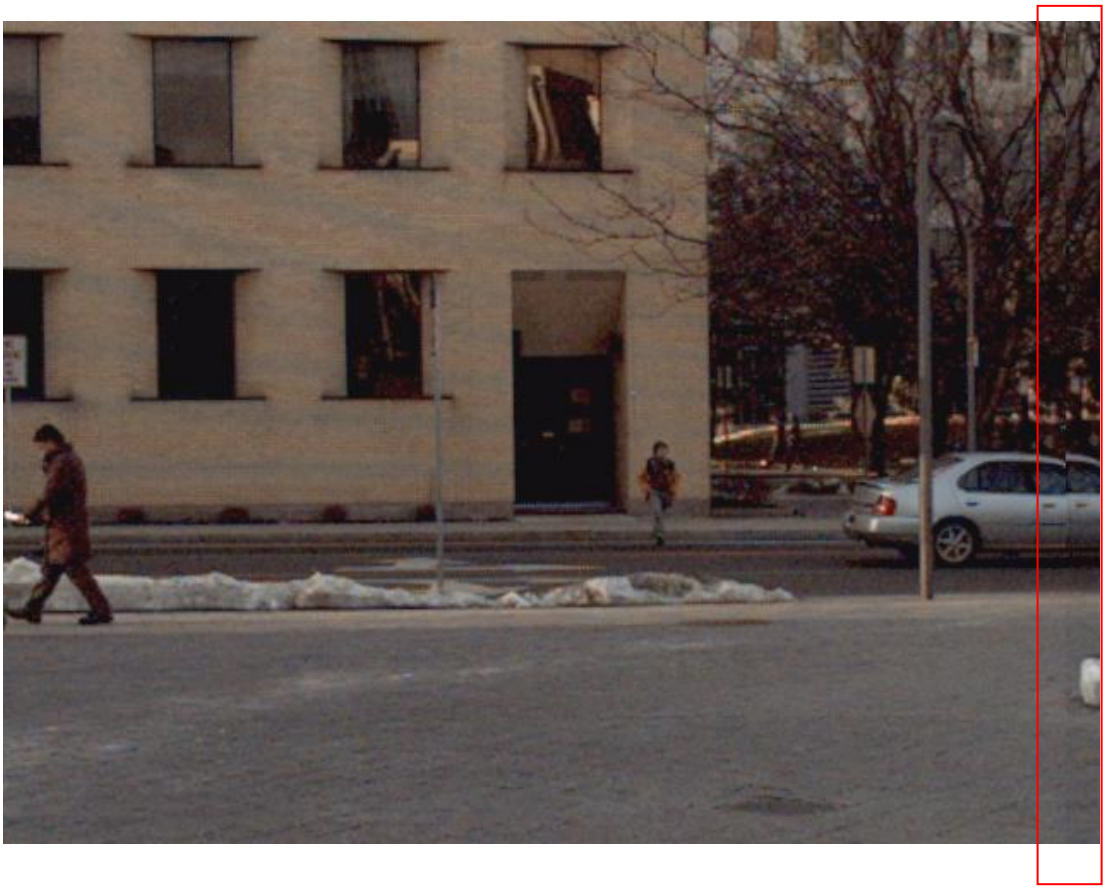


Figure 6.7: Virtual view through stitching

With a more careful look at the image, one can see that the pixels added from view 3 are not aligned with the first part of view 1. This shows that even though the cameras are vertically and horizontally aligned, the angle at which each camera captures the scene is different. Nevertheless, this simple technique can produce a virtual view at high speed at the cost of quality degradation. The overhead of the additional processing is computationally inexpensive as highlighted by Equations 1 and 2, hence a valid

candidate for reducing the transmission load on bandwidth and the number of views to be encoded but not a real representation of the original view.

6.2.2 Disparity Predicted Virtual View

The stitching method highlighted that while the cameras are aligned horizontally at equal distance from one other, each camera captures the scene from a slightly different angle. One way to find the displacement between objects, employed in stereo videos, is disparity prediction. Disparity compensation exploits the correlation between views/frames.

It has also been applied to the creation of virtual views in multi-view, albeit not for display nor at the decoder. It is used for prediction purposes. Martin et al. introduced a technique called view synthesis prediction (VSP) to improve the prediction efficiency amongst inter-views [104]. Since then, various researchers have adopted this technique to offer improvements. Flierl et al. discussed the impact of inaccurate disparity compensation for N views in a group of views (GOV) [106]. Chung et al. proposed virtual view interpolation with bilateral criterion and disparity smoothness, as an additional reference frame for encoding [107]. Hu et al. built on the idea of VSP to propose a least squared based synthesis prediction method to enhance the prediction capability of virtual view picture [108].

In this section, we employ disparity algorithm Sum of Squared Differences (SSD) to calculate the virtual view at the decoder, for the purpose of reducing the number of cameras needed to capture as many views as possible. In addition to reducing the need for cameras, the load on the encoder is reduced, as well as the bandwidth requirement

when transmitting the views and reduced storage space need. The sole aim of our investigation is to reduce the amount of data and reduce the amount of bandwidth required for transmission, and create a virtual view during the decoding process before display. Other efforts to reduce the amount of traffic for transmission still require the capturing of the virtual view and transmitting some sort of data to represent that view. Movan et al., for example, proposed replacing the transmission of the central view with its texture data and a corresponding depth map [109]. It does not absolutely eliminate the need for transmitting any information about the virtual view.

Disparity compensated prediction with SSD is usually calculated as in Equation 3 where W is the window (block) size, x and y are the coordinates of the image along x-axis and y-axis respectively and i and j are disparity locations along x-axis and y-axis.

$$\sum_{(x,y)\in W} \left(I_1(x,y) - I_2(i+x, j+y)\right)^2 \qquad \text{Equation 3.}$$

Since we know the cameras are horizontally aligned, the disparity prediction in this method is going to be just along x-axis, as represented by Equation. 4:

$$\sum_{(x,y)\in W} \left(I_1(x,y) - I_2(i+x, y)\right)^2 \qquad \text{Equation 4}$$

The cameras are placed at an equal distance from each other. The disparity between the I_1 and I_3 will be twice as much as the disparity between I_1 and I_2, and I_2 and I_3; hence the disparity is halved before applying to each half of the image to make the virtual view. The mathematical model for this process is explained as follows:

Where $I_1(x,y)$ is a frame in view 1, $I_3(x,y)$ is a frame in view 3, and $d(x,y)$ is the disparity from $I_1(x,y)$ and $I_3(x,y)$, then $d(x,y)$ from I_1I_3 will be twice as much as the required disparity for the virtual view between I_1I_2 and I_2I_3, therefore the disparity of the virtual view will be $d_H = \frac{d(x,y)}{2}$.

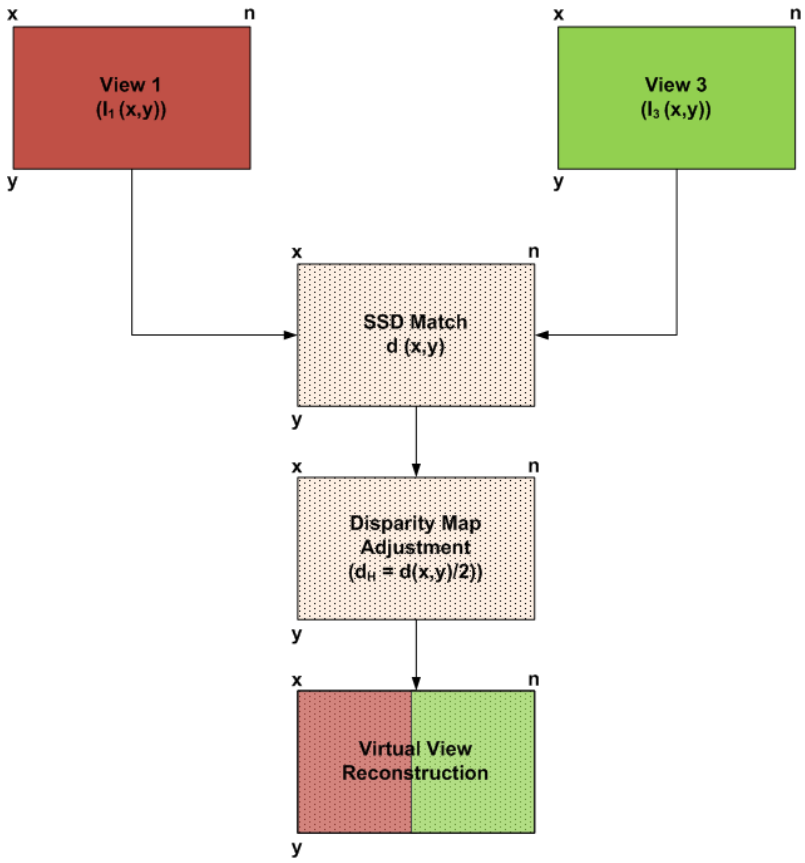


Figure 6.8: Graphical representation of the Disparity compensated virtual view creation Process

Once the disparity is calculated, the next step is to create the virtual view. The mathematical model for this two step process is represented in Equation 5 and Equation 6:

$$I_V(x,y) = I_1\left(x + d_H(x,y):\frac{n}{2}, y\right) \quad \text{Equation 5.}$$

$$I_V\left(x+\frac{n}{2}, y\right) = I_3\left((x+\frac{n}{2}) - d_H\left(x+\frac{n}{2}, y\right), y\right) \quad \text{Equation 6.}$$

In Equation 5 and Equation 6, the disparity matrix is applied first to the first half of I_1 and then to the second half of I_3. The two half images are put together in the same order to create I_V (virtual view). Figure 6.8 shows this process graphically.

Figure 6.9: Virtual view through Disparity prediction

Figure 6.9 shows the virtual view created using the above described disparity prediction method. It has definitely improved the mismatch of artifacts that were formed by the stitching process. Objective tests also show a great improvement with a PSNR of 25.6

dB which is a 3.1 dB gain from the stitching method. In addition, subjective assessment also shows more resemblance between the virtual view and the real view it was meant to reproduce. However, artifects still exist as highlighted by the green boxes. The mismatch caused by disparity errors would require further correction before making it visually closer to original view. Regardless, the results prove the viability of the method.

6.2.3 Panoramic Stitched Virtual View

Figure 6.10: Comparing the three views of Vassar for inter-view difference, where (a) is view 1, (b) is view 2 (the view we want to recreate), and (c) is view 3

Another relatively more computationally expensive method is to create the virtual view through panoramic stitching. This method uses Scale-invariant feature transform

(SIFT), RANdom SAmple Consensus (RANSAC), and Homography to generate a panoramic stitched image by using the two reference frames from View 1 and View 3. The resulting image is then cropped to the required dimensions to form the virtual view.

SIFT is used to extract keypoints in the reference images, which are then fed into RANSAC to find the outliers before applying homography to map the filtered points together. SIFT, RANSAC and Homography are well known algorithms and interested readers can refer to [110] and [111]. The basic concept and the reason for the recommendation of this method are explained below.

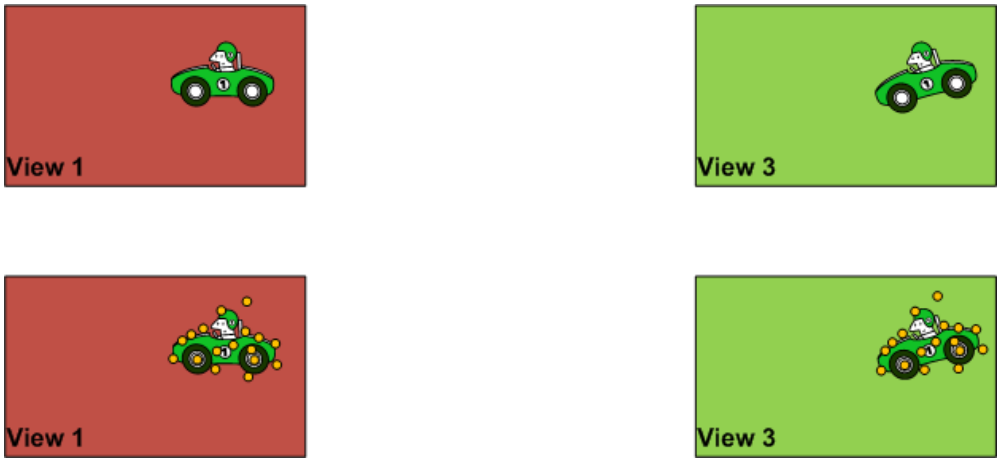


Figure 6.11: Finding keypoints in the image using sift

Figure 6.10 highlights the differences between the three views. The most prominent of which is the post sign highlighted by the red boxes. The middle view or view Figure 6.10 (b) is the one attempted to recreate here. The size of the post sign is wider than present in view 1, which is where it is hoped that the first half of the virtual view will be recreated. In view 3 (Figure 6.10 (c)), only half of the post is present. The shadow in the window, highlighted by the green boxes, is another example. In this scenario, it

shrinks moving from view 1 to 3. This is due to the angle of the camera capture, although only 19.5 cm apart, the angle at which the information is recorded makes the differences in the information captured.

These may be small visible details compared to the whole picture but it is what makes the view different. The virtual view will have to employ the changes.

Panoramic picture calculation requires the extraction of key points in an image using SIFT as depicted in Figure 6.11. The keypoints are then fed to RANSAC to find the outliers, eliminating the noise or error points as shown in Figure 6.12.

Figure 6.12: The keypoints found by SIFT are now fed to RANSAC to find the outliers highlighted by the colours red, purple and blue

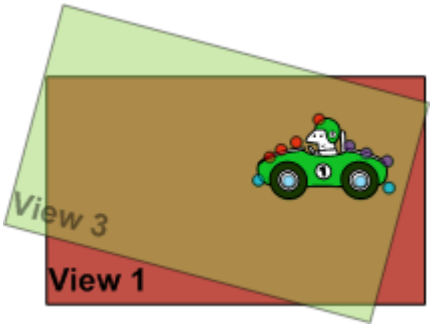


Figure 6.13: Using Homography the images are adjusted through the matching points

The outlier points are then used by the homography process to align the two views to form the panoramic view as depicted in Figure 6.13. The final virtual view is extracted by cropping the excess information from the panoramic image to form the virtual view.

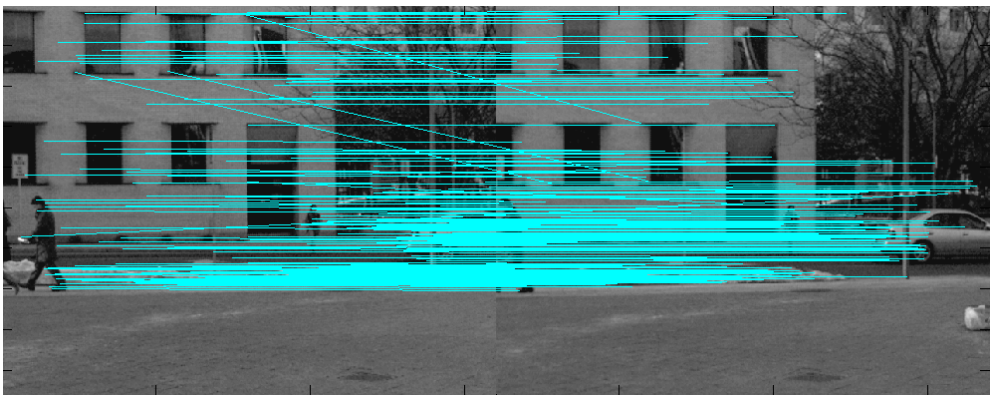


Figure 6.14: SIFT matching Points in the Vassar video sequence example using view 1 and 3 as the reference frames [111]

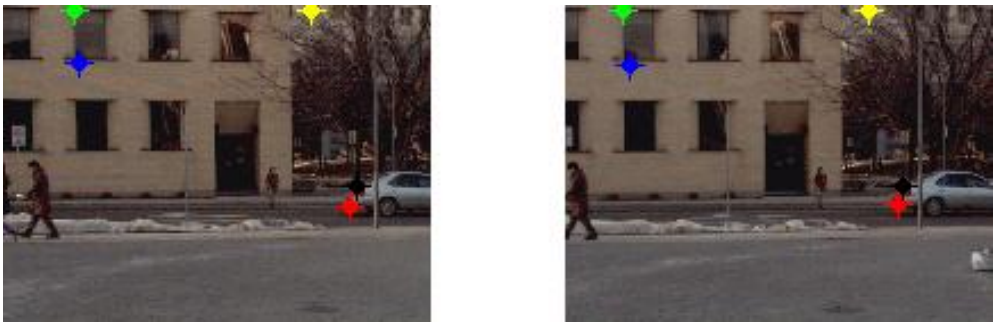


Figure 6.15: RANSAC points for the two reference images of view 1 and view 3 in the Vassar video sequence

Figure 6.14 through Figure 6.18 show this process applied to the first frame of the Vassar sequence. The final image shows some blurriness caused by the homography alignment process which could be improved but, other than that, subjective comparison with the original view shows remarkable resemblance. Objective tests, using PSNR,

show a commendable value 24 dB. It is far from perfect but a viable alternative to creating a more realistic virtual view.

Figure 6.16: Homography applied to the two images using the RANSAC points as the guide

Figure 6.17: Panoramic image created from view 1 and 3

Figure 6.18: Virtual view extracted from the panoramic image

6.3 Experimental Results

All three methods proposed in the chapter were tested on the MERL sequences, Ballroom, Vassar and Exit [111]. The sequences were encoded with JMVM 8.1 reference software for H.264/MVC. The middle view was recreated as the virtual view using views 1 and 3 as shown in Figure 6.2.

6.3.1 Computational Complexity

In order to assess the computational complexity of the three algorithms objectively, we have employed the *Big O notation*, which is also called *Landau's symbol.* It tells us how fast a function grows or declines. The letter O symbolises the rate of growth of a function, also called its order [112]. It is worth noting that *Big O notation* does not take time complexity into account. It only takes into account the growth of a function. Time complexity would depend on the processing power and coding efficiency.

To explain the order of each function, let:

- H = height of image
- W = width of image
- C = colour channels*
- D = Disparity Compensation
- R= RANSAC outlier points calculation
- F = Frame

* Three colour channels are used in our experiments, R, G and B.

Algorithm	Complexity Per Video Frame			
	Reading	Processing	Writing	Total
Stitching	O(HWC)	O(HWC)	O(HWC)	O (F) for F>0
Disparity	O(HWC)	O(HWD)	O([HWC]+D)	O (F) for F>0
Panoramic	O(HWC)	O(HWR)	O(HWC)	O (F) for F>0

Table 6.1: Complexity Calculation using Big O notation for the three algorithms

As evident from Table 6.1, the complexity of each algorithm is linear so the only function that effects growth in any of these three algorithms is the image size. Since these calculations are done for each frame the complexity will be multiplied by the number of frames.

6.3.2 Objective Quality Comparison

The average PSNR of the three methods for all three views is shown in Figure 6.19. As evident from the graph in Figure 6.19, the disparity compensated virtual view outperforms the other two. The stitching model, although in terms of time computationally the least expensive due to the number of functions it has to perform, is probably a good solution to use in error concealment, but not for creating a virtual view, because it lacks the ability to capture the scene from the right angle.

Objective results for the panoramic virtual view creation do not do it justice. Subjectively analysing, it has the most resemblance to the real scene, but unfortunately the blurriness of the view makes it structurally less similar to the original view. Frame 5, a randomly selected frame number, is displayed from Figure 6.20 through Figure 6.31 for demonstration purposes here. The frame number is kept consistent throughout the views. Figure 6.23, 6.27 and 6.31 show the virtual views created through panoramic stitching for Ballroom, Exit and Vassar respectively. Comparing their subjective quality with both the virtual views created through disparity and normal stitching models, the panoramic stitched virtual views are of a higher quality, albeit blurry. The blurriness is caused by the homography alignment. The blurriness could be improved if the homography alignment is further explored.

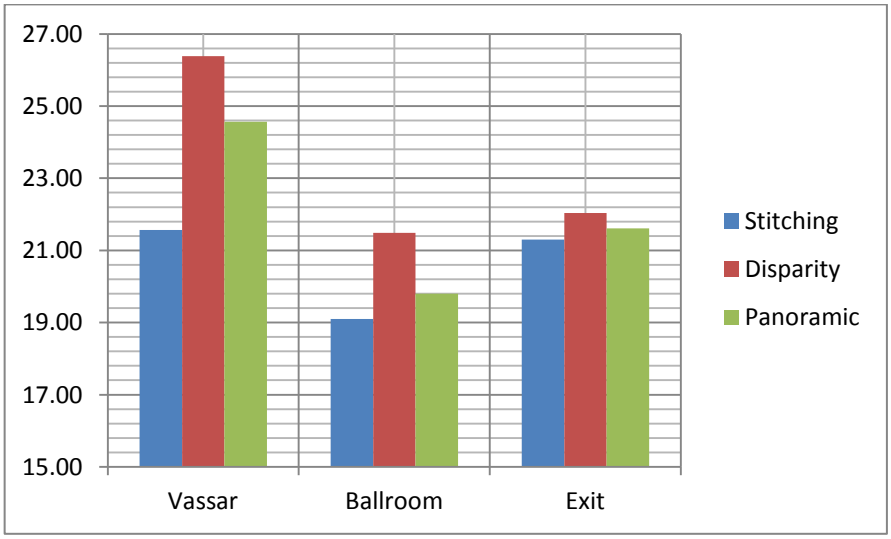


Figure 6.19: Average PSNR for the virtual view creation using the three models presented in this chapter

If there is anything that these three methods prove, it is that the creation of a virtual view at the decoder end is not only possible but computationally cheap as shown in

Table 6.1. Hence, this research proposes integrating the virtual view model into the H.264/MVC decoder. It will reduce the processing load on the encoder by reducing the amount of views to be encoded. It will further benefit the transmission channel where it will greatly reduce the amount of bandwidth required to transmit a multi-view video. With relatively less traffic on the transmission channel, one can afford to protect the transmission as proposed in Chapters 4 and 5. This process will require the decoder to have more processing power than previously required because of the additional rendering of the virtual view. This, however, should not be a problem since most end points, from smart phones to personal computers, have in recent years been equipped with enhanced processing capabilities that could be used to assist with reducing the delivery load required for multi-view videos.

6.3.3 Subjective Quality Comparison

Figure 6.20: Original frame 5 of the Ballroom sequence

Figure 6.21: Virtual Frame 5 of the Ballroom sequence created through stitching

Figure 6.22: Virtual frame 5 of the Ballroom sequence created through disparity compensated prediction

Figure 6.23: Virtual frame 5 of the Ballroom sequence created through panoramic stitching

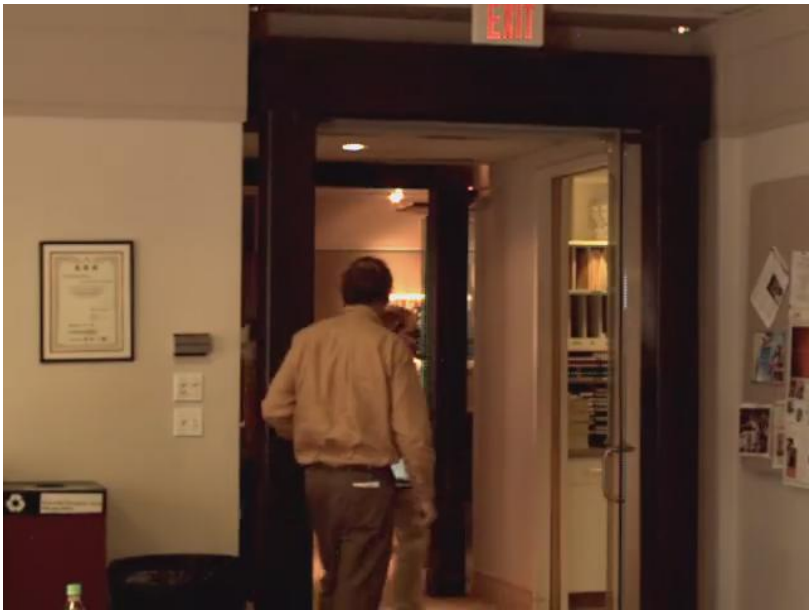


Figure 6.24: Original frame 5 of the Exit sequence

Figure 6.25: Virtual frame 5 of the Exit sequence created through stitching

Figure 6.26: Virtual frame 5 of the Exit sequence created through disparity compensated prediction

Figure 6.27: Virtual frame 5 of the Exit sequence created through panoramic stitching

Figure 6.28: Original frame 5 of the Vassar sequence

Figure 6.29: Virtual frame 5 of the Vassar sequence created through stitching

Figure 6.30: Virtual frame 5 of the Vassar sequence created through disparity compensated prediction

Figure 6.31: Virtual frame 5 of the Vassar sequence created through panoramic stitching

6.4 Analysis and Discussion

Multi-view videos are captured with synchronised cameras that are horizontally aligned with an equal distance between them. It is therefore possible to extract a virtual view by taking one adjacent camera image on each side. The similarities between the frames are already exploited in multi-view video coding (MVC) extension of H.264 advanced video coding (AVC) during compression. The middle view is bi-predicted using the frames from the adjacent views. The bi-prediction improves coding efficiency and reduces the amount of data but does not completely eliminate the middle view.

The experimental results shown above prove that if one out of three cameras can be eliminated completely. The middle view can be recreated during the decoding process in real time. In addition to reducing the number of cameras required to capture a multi-

view video, virtual view rendering will further improve the efficiency of the codec. This will effectively reduce the bandwidth requirement for multi-view video transmission. However, rendering virtual view does not come without challenges. The first and the most important challenge is occlusion, where certain elements in the visual scene are seen by one camera but not the other due to differences in cameras viewpoint. The rendered virtual view must be structurally similar to the original view it aims to imitate.

Three methods are presented based on their computational simplicity (see Table 6.1 for details) to ensure virtual view rendering can be done in real time. The first method which employs cropping excess pixels from both views and then stitching the remaining to form the virtual view is simplest of all but it cannot handle the occlusion problem. A multi-view video offers viewers different perspective views from different angles. While this method includes all the information that would be included in the original view, the angle at which it is captured is not the same. Hence, it may only be used in error concealment but not an advisable contender for virtual view rendering for large number of frames.

The disparity compensated prediction and panoramic image stitching are the two real contenders that can render a virtual view similar to the original view. Disparity compensated view rendering is a more traditional method because of its wide use in image processing, not for virtual view rendering but for other processes such as the Depth Based Image Rendering (DBIR). It is quick to implement but cannot handle occlusions very well.

The panoramic image stitching is not a new concept. As the name suggests it is used to stitch several images of a scene to create a panoramic view. From the author's literature survey, there was no evidence of it being used for virtual view rendering. The benefit of using panoramic image stitching is because it goes beyond checking the similarities between the frames being considered. It also aligns them based on a number of matching points so the angle of the image is adjusted to create a more realistic virtual view. It therefore handles occlusions better and the resulting view is more similar to the original view. This is evident from both subjective and objective testing performed as shown in the experimental results.

Including the virtual view rendering process as part of the MVC decoder will improve coding efficiency as highlighted in this chapter and reduce the bandwidth requirement for transmission. It will however increase the workload on the decoder. With end-devices today having more processor power than servers that were available a decade ago, this additional work should not be a challenge. The other advantage of using virtual view rendering, in addition to reducing capturing costs, improving coding efficiency and reducing bandwidth requirement for transmission, is to balances the load between the encoder and the decoder. For real-time transmission reducing the encoding time is a critical component of coding efficiency.

6.5 Conclusion

This chapter suggests decoder optimisation in H.264 Multi-view video coding through the insertion of full resolution virtual views from neighbouring decoded views. Three methods are proposed, namely image stitching, disparity compensated prediction, and panoramic stitching, to create virtual views using two reference views from the decoded bitstream. The advantages and shortages of each process are discussed with visual aid and objective results. The shortcomings of the objective methods to correctly analyse the images are also highlighted.

All three methods are computationally inexpensive, requiring minor manipulations, and mathematical models which are presented in this chapter. The viability of considering two of three models for creating virtual views is recommended whereas the third method, image stitching, can be employed for error correction in the middle views, in case of frame loss.

Disparity prediction requires less computational load than the panoramic view but it is not able to overcome occlusions where an object in the scene is smaller in the middle view because of the angle it was captured from in the reference views. The panoramic stitching model, however, is relatively more computationally expensive. It also suffers from blurriness due to the incorrect alignment during the homography process. But subjectively speaking, it is the method which produces the best result. With further study of the homography alignment, it could produce virtual views almost identical to the original view.

This thesis has so far discussed the challenges in 3D video communication technologies, compression of 3D videos, transmission, and virtual view rendering in multi-view video transmissions. The next chapter concludes the thesis and highlights the key achievements of this research.

Chapter 7: Conclusions and Future Work

This chapter concludes the research contributions in this thesis and the recommendations made in the preceding chapters. The aim of this research was to find solutions to optimise the end-to-end 3D video communication system. A list of tasks for future work is also recommended in this chapter.

7.1 Overview

3D video communication solutions must consider the entire pipeline, starting with optimisation at the video source to the end display and transmission optimisation. *Multi-view* offers the most compelling solution for 3D videos with motion parallax and freedom from wearing headgear for 3D video perception. Optimising multi-view video for delivery and display could increase the demand for true 3D in the consumer market.

There are several challenges still facing the 3D systems, ranging from *content creation*, *error resilience* and *concealment*, *heterogeneity*, achieving true *free-view-point TV*, and *quality of experience*. This thesis addresses some of these challenges and recommends solutions that would optimise the quality of 3D videos throughout the communication pipeline in heterogeneous networks.

In **chapter 1**, an overview of the trends in 3D video technological advancements was presented. The underlying *techniques in 3D video perceptions* were discussed followed by the way they are represented in videos. The fact that different representations affect the video perception depending on the compromises made to represent them was highlighted. The most ideal representation is a scenario where all views are used to the

fullest extent. This makes for the best 3D perception but increases the amount of data that needs to be recorded proportionate to the number of views captured.

3D displays correlating to the 3D video representation were also discussed, including those that are still under current investigation in research. The most promising display that could increase the uptake of the 3D video displays but the general public is the *Multiview Autostereoscopic 3D*. The challenge with multiview displays is the requirement to capture the different views using *2D video cameras*, which need to be *synchronised and calibrated*. But the biggest challenge is to optimise them in order for delivery over the internet and display.

7.2 Optimisation of 3D videos through Compression

Multi-view video coding enables auto-stereoscopic displays and Free-view-point TV for the mass market. It is the most promising technology in 3D videos to date. But it requires a large amount of data to be captured by multiple synchronised 2D cameras. The similarities between the views can be exploited for *coding efficiency*. Multi-view video coding (MVC) extension of the H.264/AVC standard was explained and its benefits highlighted in **chapter 3**.

A comparison of video compression on 3D video is presented taking 2D video as a benchmark. Two views are encoded with the stereo profile of the *Advanced Video Coding (AVC) standard*, first with treating each view as an independent view and then by exploiting the *inter-view dependency*. The results show that while there is no considerable quality degradation, the *savings in bitrate can range from 3% to 24%* in the test bed used in the experiments. The bitrate analysis presented in this work was to

demonstrate the range of bit rate savings when two views are compressed in a stereoscopic encoding scheme as compared to *simulcast compression*. The stereoscopic coding scheme can be exploited further to reduce the bitrate by incorporating other techniques such as *depth map estimation*.

The results shown in **chapter 3** demonstrated the importance of selecting the correct codec for *robust compression efficiency*. The stereo profile of the H.264/AVC, where the redundancies between the views are exploited without any considerable effect on the quality of the video, highlight how efficiency in 3D video can be achieved, which provides the bases for using MVC as the standard codec for 3D videos.

7.3 Optimising the Transmission of 3D Videos

Transmission of the *multi-view video* requires more bandwidth compared to its *spatially equivalent 2D counterpart*. The reason for this is the increased number of views. H.264 MVC can encode videos with better efficiency by exploiting *inter-view redundancy* but with considerably more data to transmit.

In a network where quality is an important factor, the video can be transmitted via HTTP which is built on top of TCP/IP at the cost of extra time. It guarantees the delivery and benefits from security clearances as it is not blocked by most firewalls. However, time is critical in the transmission. **Chapter 4** explored the idea of *varying packet sizes* to reduce the delivery time. Small packets are favoured by real-time video streaming and they are also used in mobile networks but they require more processing power and are more prone to loss in congested networks.

Large packet sizes take longer and if lost can affect the video but since in TCP it will be retransmitted, they are preferred. Moreover, the largest packet size that a network can handle is dictated by the *Maximum Transmission Unit (MTU)* of the weakest link in the network. MTU differs for each network. The experiments in this work used Ethernet with an *MTU size* of 1500 bytes.

Application designers should consider dynamic path discovery to understand the largest packet sizes allowed avoiding fragmentation of packets en route and the processing power of the end-devices. *Increased amount of packets* will increase the required *processing power*.

MVC greatly reduces the amount of data that needs to be transmitted by exploiting inter-view redundancies between *adjacent video sequences* in addition to the regular temporal and spatial redundancies in each view. This, however, makes it more *vulnerable to transmission errors*. Errors in key frames can render the entire bitstream *un-decodable*.

The inter-view dependency also means that errors propagate to other views when errors occur in frames that are used by other views for motion compensation prediction. **Chapter 5** investigated the transmission of MVC bitstreams over multiple channels based on priority, and view changes to improve the video quality.

MST gives the end-devices the option to drop streams in congested networks in order to allocate more bandwidth to the base view for better QoE. MST can also be used to reap the advantages of *simulcast* without losing the benefits of *inter-view similarities* exploitation. It does, however, introduce problems with the synchronisation in

erroneous channel conditions and when packets arrive out of bounds. This problem has been resolved by using the *frame order* to resynchronise by taking advantage of the MVC *time-first frame ordering*.

7.4 Decoder Optimisation for 3D Videos

Chapter 6 suggested decoder optimisation in H.264 Multi-view video coding through the insertion of full resolution virtual views from neighbouring decoded views. Three methods were proposed, namely *image stitching*, *disparity compensated prediction*, and *panoramic stitching* to create *virtual views* using two reference views from the decoded bitstream. The advantages and shortages of each process were discussed with visual aid and objective results. The shortcomings of the objective methods to correctly analyse the images is also highlighted.

All three methods are computationally inexpensive; require minor manipulations, mathematical models for which are presented in **chapter 6**. The viability of considering two of three models for creating virtual views is recommended where as the third method, image stitching, can be employed for *error correction* in middle views, in case of frame loss.

Disparity prediction requires less computational load than the panoramic view but it is not able to overcome occlusions and where an object in the scene is smaller in the middle view because of the angle it was captured from in the reference views. The panoramic stitching model, however, is relatively more computationally expensive. It also suffers from blurriness due to the incorrect alignment during the *homography* process. But subjectively speaking, it is the method which produces the best result.

With further study of the homography alignment, it could produce virtual views almost identical to the original view.

7.5 Future Work

Quality optimisation in 3D video communication will be an *ongoing research area* for the foreseeable future. The findings presented in this research investigate the end-to-end challenges in the communication pipeline. They are by no means an end result. Each section of this research could be further investigated. The following key future work areas are recommended for future research uptake:

I. MVC decoder fails when there is an error in the bitstream. One way to address this issue would be to make the codec more robust through *synchronisation* points using *Instantaneous Decoder Refresh (IDR)* pictures. When an error occurs, the codec can simply ignore the rest of the non-decodable bitstream until the next IDR picture is received. It certainly is not the ultimate solution but is a good start at making the *JMVC* reference software more *robust*.

II. Further investigate virtual view rendering using the panoramic view stitching and disparity compensated view prediction to improve the quality of the virtual view created from the decoded adjacent views in multi-view video coding.

III. Add virtual view rendering to the *decoder*. The virtual view rendering techniques presented in this research could be used as a starting point to modifying the decoder.

IV. Investigate *HTTP and RTP combined transmission* for multi-view video delivery, where the base view and inter-view predicted frames in a group of

pictures are transmitted using HTTP and the inter-view bi-predicted frames through RTP. The cost of such a transmission needs to be evaluated in terms of time and the quality improvements.

References

[1]. Tekalp, A. MURAT, Aljoscha Smolic, Anthony Vetro, and Levent Onural. "Special issue on 3-D media and displays." *Proceedings of the IEEE* 99, no. 4 (2011): 536-539.

[2]. Caviedes, Jorge E. "The Evolution of Video Processing Technology and Its Main Drivers." *Proceedings of the IEEE* 100, no. 4 (2012): 872-877.

[3]. Kunic, Srecko, and Zoran Sego. "Beyond HDTV technology." In *ELMAR, 2013 55th International Symposium*, pp. 83-87. IEEE, 2013.

[4]. Wang, Yao, Jörn Ostermann, and Ya-Qin Zhang. *Video processing and communications*. Vol. 5. Upper Saddle River: Prentice Hall, 2002.

[5]. Ohtsuka, Satoko, and Shinya Saida. "Depth perception from motion parallax in the peripheral vision." In *Robot and Human Communication, 1994. RO-MAN'94 Nagoya, Proceedings., 3rd IEEE International Workshop on*, pp. 72-77. IEEE, 1994.

[6]. Matusik, Wojciech, and Hanspeter Pfister. "3D TV: a scalable system for real-time acquisition, transmission, and autostereoscopic display of dynamic scenes." In *ACM Transactions on Graphics (TOG)*, vol. 23, no. 3, pp. 814-824. ACM, 2004.

[7]. Johanson, Mathias. "Stereoscopic video transmission over the Internet." In*Internet Applications, 2001. WIAPP 2001. Proceedings. The Second IEEE Workshop on*, pp. 12-19. IEEE, 2001.

[8]. Dubois, Eric. "A projection method to generate anaglyph stereo images." In*Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*, vol. 3, pp. 1661-1664. IEEE, 2001.

[9]. Toperverg, B., O. Nikonov, V. Lauter-Pasyuk, and H. J. Lauter. "Towards 3D polarization analysis in neutron reflectometry." *Physica B: Condensed Matter*297, no. 1 (2001): 169-174.

[10]. Brian, Marshall, How 3-D Glasses Work, Available at: http://science.howstuffworks.com/3-d-glasses2.htm [Last accessed on 20 January 2014]

[11]. Divelbiss, Adam W., David C. Swift, and Walter V. Tserkovnyuk. "3D stereoscopic shutter glass system." *U.S. Patent Application 10/764,277*, filed January 23, 2004.

[12]. Stelter, Brian, and Stone, Brad, Television Begins a Push Into the 3rd Dimension, Available at: http://www.nytimes.com/2010/01/06/business/media/06tele.html [last accessed on 20 January 2014]

[13]. Su, Guan-Ming, Yu-Chi Lai, Andres Kwasinski, and Haohong Wang. "3D video communications: Challenges and opportunities." *International Journal of Communication Systems* 24, no. 10 (2011): 1261-1281.

[14]. Urey, Hakan, Kishore V. Chellappan, Erdem Erden, and Phil Surman. "State of the art in stereoscopic and autostereoscopic displays." *Proceedings of the IEEE*99, no. 4 (2011): 540-555.

[15]. Vetro, Anthony, Alexis M. Tourapis, Karsten Muller, and Tao Chen. "3D-TV content storage and transmission." *Broadcasting, IEEE Transactions on* 57, no. 2 (2011): 384-394.

[16]. Tikanmaki, A., A. Gotchev, A. Smolic, and K. Miller. "Quality assessment of 3D video in rate allocation experiments." In *Consumer Electronics, 2008. ISCE 2008. IEEE International Symposium on*, pp. 1-4. IEEE, 2008.

[17]. Smolic, Aljoscha, Karsten Mueller, Nikolce Stefanoski, Joern Ostermann, Atanas Gotchev, Gözde B. Akar, Georgios Triantafyllidis, and Alper Koz. "Coding algorithms for 3DTV—a survey." *Circuits and Systems for Video Technology, IEEE Transactions on* 17, no. 11 (2007): 1606-1621.

[18]. Chen, Ying, Ye-Kui Wang, Kemal Ugur, Miska M. Hannuksela, Jani Lainema, and Moncef Gabbouj. "The emerging MVC standard for 3D video services."*EURASIP Journal on Applied Signal Processing* 2009 (2009): 8.

[19]. Currys, Showing results for "3D TV", available at: http://www.currys.co.uk/gbuk/search-keywords/xx_xx_xx_xx_xx/3d+tv/xx-criteria.html [Last Accessed on 10 January 2014]

[20]. Carter, Jamie, 3D TV: all your questions answered, available at: http://www.techradar.com/news/television/3d-tv-all-your-questions-answered-987535, [Last Accessed on 10 January 2014]

[21]. Ackerman, Dan, Is the glasses-free 3D laptop ready for prime time?, Available at: http://reviews.cnet.com/8301-3121_7-57428025-220/is-the-glasses-free-3d-laptop-ready-for-prime-time/ [Last Accessed on: 10 January 2014]

[22]. Tridelity 3D Technologies, "3D Technologies, Displays & 3D Creation from Tridelity 3D Technologies", available online at: http://www.tridelity.com/Technology.3d-display.0.html [Last Accessed on 10 January 2014]

[23]. Su, Guan-Ming, Yu-Chi Lai, Andres Kwasinski, and Haohong Wang. "3D Display Systems." *3D Visual Communications*: 63-84.

[24]. Olsson, Roger. "Synthesis, Coding, and Evaluation of 3D Images Based on Integral Imaging." PhD diss., LNU, 2008.

[25]. Available online at: http://www.udel.edu/biology/Wags/wagart/anaglyphpage/buckyball.gif [Last Accessed on 10 January 2014]

[26]. 3D Vision Blog, "Using Different Anaglyph Glasses with GeForce 3D Vision" , Available online at: http://3dvision-blog.com/117-using-different-anaglyph-glasses-with-geforce-3d-vision/ [Last Accessed on 10 January 2014]

[27]. Boher, Pierre, Thierry Leroux, Thibault Bignon, and Véronique Collomb-Patton. "Multispectral polarization viewing angle analysis of circular polarized stereoscopic 3D displays." *Electronic Imaging, San Jose, Proc. SPIE* 7524 (2010): 26.

[28]. Baker, Simon, "3D Display Technologies", Available online at: http://www.tftcentral.co.uk/articles/3d_technologies.htm [Last Accessed on 10 January 2014]

[29]. Sony, "Personal 3D Viewer", Available online at: http://www.sony.co.uk/electronics/head-mounted-display-products/hmz-t2 [Last Accessed on 10 January 2014]

[30]. Lewis, Jordan D., Carl M. Verber, and Robert B. McGhee. "A true three-dimensional display." *Electron Devices, IEEE Transactions on* 18, no. 9 (1971): 724-732.

[31]. Dodgson, Neil A. "Autostereoscopic 3D displays." *Computer 38, no. 8 (2005)*: 31-36.

[32]. Yendo, Tomohiro, Toshiaki Fujii, Masayuki Tanimoto, and Mehrdad Panahpour Tehrani. "The Seelinder: Cylindrical 3D display viewable from 360 degrees."*Journal of visual communication and image representation* 21, no. 5 (2010): 586-594.

[33]. Strauss, Paul, "Sonny 360-degree volumetric display: the stuff sci-fi dreams are made of", Available online at: http://technabob.com/blog/2010/07/19/sony-360-degree-volumetric-display/ [Last Accessed on 10 January 2014]

[34]. Gregg, E. "Volumetric 3D displays and application infrastructure." *IEEE Computer Society* 8 (2005): 37-44.

[35]. Hill, P. C. J. "Dennis Gabor-Contributions to Communication Theory & Signal Processing." *In EUROCON, 2007. The International Conference on" Computer as a Tool"*, pp. 2632-2637. IEEE, 2007.

[36]. Weber, David C., and James D. Trolinger. "Holographic labeling and reading machine for authentication and security applications." *U.S. Patent 5,920,058*, issued July 6, 1999.

[37]. Benton, Stephen A. "Holographic Displays: 1975-1980." *Optical Engineering* 19, no. 5 (1980): 195686-195686.

[38]. Casper, J. E., and S. A. Feller. "The Complete Hologram Book." (1987).

[39]. Saxby, Graham. *Practical holography*. CRC Press, 2010.

[40]. Hariharan, Parameswaran. *Basics of holography*. Cambridge University Press, 2002.

[41]. The Naledi3d Factory, “3D and Holographic Display Solutions”, Available online at: http://www.naledi3d.com/EON_3d.html [Last Accessed on 10 January 2014]

[42]. Aggoun, Amar, Emmanuel Tsekleves, Mohammad Rafiq Swash, Dimitrios Zarpalas, Anastasios Dimou, Petros Daras, Paulo Nunes, and Luis Ducla Soares. "Immersive 3D holoscopic video system." *MultiMedia, IEEE* 20, no. 1 (2013): 28-37.

[43]. Aggoun, Amar. "3D holoscopic imaging technology for real-time volume processing and display." In *High-Quality Visual Experience*, pp. 411-428. Springer Berlin Heidelberg, 2010..

[44]. Ng, Ren, Marc Levoy, Mathieu Brédif, Gene Duval, Mark Horowitz, and Pat Hanrahan. "Light field photography with a hand-held plenoptic camera."*Computer Science Technical Report CSTR* 2, no. 11 (2005).

[45]. Aggoun, Amar. "3D Holoscopic video content capture, manipulation and display technologies." In *Information Optics (WIO), 2010 9th Euro-American Workshop on*, pp. 1-3. IEEE, 2010.

[46]. Swash, M. Rafiq, A. Aggoun, O. Abdulfatah, B. Li, J. C. Fernandez, and E. Tsekleves. "Holoscopic 3D image rendering for autostereoscopic multiview 3D display." In *Broadband Multimedia Systems and Broadcasting (BMSB), 2013 IEEE International Symposium on*, pp. 1-4. IEEE, 2013.

[47]. Okoshi, Takanori. *Three-dimensional imaging techniques*. Academic Press, 1976.

[48]. Ives, Herbert E. "Lenticulated sheet." (1931).

[49]. McCormick, M., and N. Davies. "Full natural colour 3D optical models by integral imaging." In *Holographic Systems, Components and Applications, 1993., Fourth International Conference on*, pp. 237-242. IET, 1993.

[50]. Okano, Fumio, Jun Arai, Haruo Hoshino, and Ichiro Yuyama. "Real-time three-dimensional pickup and display system based on Integral Photography." In*SPIE's International Symposium on Optical Science, Engineering, and Instrumentation*, pp. 70-79. International Society for Optics and Photonics, 1998.

[51]. Okano, Fumio, Jun Arai, Haruo Hoshino, and Ichiro Yuyama. "Three-dimensional video system based on integral photography." *Optical Engineering* 38, no. 6 (1999): 1072-1077.

[52]. Davies, Neil, Malcolm McCormick, and Li Yang. "Three-dimensional imaging systems: a new development." *Applied optics* 27, no. 21 (1988): 4520-4528.

[53]. Davies, Neil A., Michael Brewin, and Malcolm McCormick. "Design and analysis of an image transfer system using microlens arrays." *Optical Engineering* 33, no. 11 (1994): 3624-3633.

[54]. McCormick, M., and N. Davies. "Full natural colour 3D optical models by integral imaging." In *Holographic Systems, Components and Applications, 1993., Fourth International Conference on*, pp. 237-242. IET, 1993.

[55]. Zinger, Svitlana, Luat Do, Daniel Ruijters, and Peter HN de With. "iglance: Interactive free viewpoint for 3d tv." In *17th International Conference on Computer Graphic, Visualization and Computer Vision*. 2010.

[56]. Funk, Walter. "History of autostereoscopic cinema." In *IS&T/SPIE Electronic Imaging*, pp. 82880R-82880R. International Society for Optics and Photonics, 2012.

[57]. Tetsutani, Nobuji, Katsuyuki Omura, and Fumio Kishino. "Wide-screen autostereoscopic display system employing head-position tracking." *Optical Engineering* 33, no. 11 (1994): 3690-3697.

[58]. Johnson, R. Barry, and Gary A. Jacobsen. "Advances in lenticular lens arrays for visual display." In *Optics & Photonics 2005*, pp. 587406-587406. International Society for Optics and Photonics, 2005.

[59]. Saveljev, Vladimir V. "Characteristics of moiré spectra in autostereoscopic three-dimensional displays." *Display Technology, Journal of* 7, no. 5 (2011): 259-266.

[60]. Van Berkel, Cornelis, and John A. Clarke. "Autostereoscopic display apparatus." *U.S. Patent 6,064,424*, issued May 16, 2000.

[61]. Takaki, Yasuhiro, Osamu Yokoyama, and Goro Hamagishi. "Flat panel display with slanted pixel arrangement for 16-view display." In *IS&T/SPIE Electronic Imaging*, pp. 723708-723708. International Society for Optics and Photonics, 2009.

[62]. Alioscopy 3D display pixel mapping principles, Available online at http://www.alioscopy.com/en/principles.php [Last Accessed on 10 January 2014]

[63]. Son, Jung-Young, Bahram Javidi, and Kae-Dal Kwack. "Methods for displaying three-dimensional images." *Proceedings of the IEEE* 94, no. 3 (2006): 502-523.

[64]. Ultra-D, Available online at: http://www.streamtvnetworks.com/ultra-d.shtml [Last Accessed on 10 January 2014]

[65]. Fisher, Will, Martin Suchara, and Jennifer Rexford. "Greening backbone networks: reducing energy consumption by shutting off cables in bundled links." In *Proceedings of the first ACM SIGCOMM workshop on Green networking*, pp. 29-34. ACM, 2010.

[66]. Onural, Levent, T. Sikora, J. Ostermann, A. Smolic, M. Reha Civanlar, and John Watson. "An assessment of 3DTV technologies." In *Proceedings of the NAB Broadcast Engineering Conference*, pp. 456-467. 2006.

[67]. Park, Young Kyung, Kwanghee Jung, Youngjin Oh, Suyoung Lee, Joong Kyu Kim, Gwangsoon Lee, Hyun Lee, Kugjin Yun, Namho Hur, and Jinwoong Kim. "Depth-image-based rendering for 3DTV service over T-DMB." *Signal Processing: Image Communication* 24, no. 1 (2009): 122-136.

[68]. Gotchev, Atanas, A. Tikanmaki, Atanas Boev, Karen Egiazarian, Ivan Pushkarov, and Nikolai Daskalov. "Mobile 3DTV technology demonstrator based on OMAP 3430." In *Digital Signal Processing, 2009 16th International Conference on*, pp. 1-6. IEEE, 2009.

[69]. Bici, M. Oguz, Döne Bugdayci, Gozde Bozdagi Akar, and Atanas Gotchev. "Mobile 3D video broadcast." In *Image Processing (ICIP), 2010 17th IEEE International Conference on*, pp. 2397-2400. IEEE, 2010.

[70]. Fehn, C., P. Kauff, M. Op De Beeck, F. Ernst, W. Ijsselsteijn, M. Pollefeys, L. Van Gool, E. Ofek, and I. Sexton. "An evolutionary and optimised approach on 3D-TV." *In Proc. of IBC*, vol. 2, pp. 357-365. 2002.

[71]. Hur, Namho, Hyun Lee, Gwang Soon Lee, Sang Jin Lee, Atanas Gotchev, and Sang-Il Park. "3DTV broadcasting and distribution systems." *Broadcasting, IEEE Transactions on* 57, no. 2 (2011): 395-407.

[72]. Lamboray, Edouard, Stephan Wurmlin, and Markus Gross. "Real-time streaming of point-based 3D video." In *Virtual Reality, 2004. Proceedings. IEEE*, pp. 91-281. IEEE, 2004.

[73]. Shi, Shu, Won J. Jeon, Klara Nahrstedt, and Roy H. Campbell. "Real-time remote rendering of 3D video for mobile devices." In *Proceedings of the 17th ACM international conference on Multimedia*, pp. 391-400. ACM, 2009.

[74]. Aksay, Anil, Selen Pehlivan, Engin Kurutepe, Cagdas Bilen, Tanir Ozcelebi, Gozde Bozdagi Akar, M. Reha Civanlar, and A. Murat Tekalp. "End-to-end stereoscopic video streaming with content-adaptive rate and format control."*Signal Processing: Image Communication* 22, no. 2 (2007): 157-168.

[75]. Xin, Baicheng, Ronggang Wang, Zhenyu Wang, Wenmin Wang, Chenchen Gu, Quanzhan Zheng, and Wen Gao. "AVS 3D video streaming system over internet." In *Signal Processing, Communication and Computing (ICSPCC), 2012 IEEE International Conference on*, pp. 286-289. IEEE, 2012.

[76]. Riechert, Christian, Frederik Zilly, Peter Kauff, Jens Güther, and Ralf Schäfer. "Fully automatic stereo-to-multiview conversion in autostereoscopic displays." In*Proc. IBC*, pp. 8-14. 2012.

[77]. Lai, Yeong-Kang, Yu-Fan Lai, and Ying-Chang Chen. "An Effective Hybrid Depth-Generation Algorithm for 2D-to-3D Conversion in 3D Displays." *Journal of Display Technology* 9, no. 3 (2013): 154-161.

[78]. Fatah, O. Abdul, A. Aggoun, M. Nawaz, J. Cosmas, E. Tsekleves, M. Rafiq Swash, and E. Alazawi. "Depth mapping of integral images using a hybrid disparity analysis algorithm." In *Broadband Multimedia Systems and Broadcasting (BMSB), 2012 IEEE International Symposium on*, pp. 1-4. IEEE, 2012.

[79]. ITU-T Recommendation, "H.264 – Advanced video coding for generic audiovisual services", Jan 2012

[80]. Vetro, Anthony, 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* 99, no. 4 (2011): 626-642.

[81]. ftp://ftp.merl.com/pub/avetro/mvc-testseq

[82]. Schwarz, Heiko, and Thomas Wiegand. "Inter-view prediction of motion data in multiview video coding." In *Picture Coding Symposium (PCS), 2012*, pp. 101-104. IEEE, 2012.

[83]. Skupin, Robert, Peiyu Yue, Thomas Schierl, and Ye-Kui Wang. "RTP Payload Format for MVC Video." (2012).

[84]. Schierl, Thomas, and Sam Narasimhan. "Transport and storage systems for 3-D video using MPEG-2 systems, RTP, and ISO file format." *Proceedings of the IEEE* 99, no. 4 (2011): 671-683.

[85]. Wiegand, Thomas, Gary J. Sullivan, Gisle Bjontegaard, and Ajay Luthra. "Overview of the H. 264/AVC video coding standard." *Circuits and Systems for Video Technology, IEEE Transactions on* 13, no. 7 (2003): 560-576.

[86]. Murakami, Tomokazu, Shohei Saito, Yuto Komatsu, Katsuyuki Nakamura, and Toru Yokoyama. "Enhancement of H. 264/AVC for higher coding efficiency

using motion estimation between reference frames." *Consumer Electronics, IEEE Transactions on* 56, no. 2 (2010): 925-929.

[87]. Sullivan, Gary J., Jens Ohm, Woo-Jin Han, and Thomas Wiegand. "Overview of the high efficiency video coding (HEVC) standard." *Circuits and Systems for Video Technology, IEEE Transactions on* 22, no. 12 (2012): 1649-1668.

[88]. Milovanovic, Dragorad, and Zoran Bojkovic. "MPEG Video deployment in interactive multimedia systems: HEVC vs. AVC codec performance study." *WSEAS Transactions on Signal Processing* 9, no. 4 (2013)

[89]. Martinian, Emin, Alexander Behrens, Jun Xin, Anthony Vetro, and Huifang Sun. "Extensions of H. 264/AVC for multiview video compression." In *Image Processing, 2006 IEEE International Conference on*, pp. 2981-2984. IEEE, 2006.

[90]. Mohorko, Jože, Matjaž Fras, and Zarko Cucej. "Real video stream transmission over simulated wireless link." In *Advanced Technologies for Communications, 2008. ATC 2008. International Conference on*, pp. 231-234. IEEE, 2008.

[91]. Kordelas, Athanasios, Tasos Dagiuklas, and Ilias Politis. "On the performance of H. 264/MVC over lossy IP-based networks." In *Signal Processing Conference (EUSIPCO), 2012 Proceedings of the 20th European*, pp. 1149-1153. IEEE, 2012.

[92]. ftp://ftp.merl.com/pub/avetro/mvc-testseq/orig-yuv, [Last accessed on 25 Sept 2012]

[93]. OPNET 17.1 PL0 Documentation

[94]. Wenger, Stephan. "H. 264/AVC over IP." *Circuits and Systems for Video Technology, IEEE Transactions on* 13, no. 7 (2003): 645-656.

[95]. J Mogul & Steve Deering, Path MTU Discovery, *RFC 1191*, (1990)

[96]. Mathis, M. and J. Heffner, "Packetization Layer Path MTU Discovery", *RFC 4821*, (2007)

[97]. Gurler, CGoektug, Burak Gorkemli, Gorkem Saygili, and A. Murat Tekalp. "Flexible transport of 3-D video over networks." *Proceedings of the IEEE* 99, no. 4 (2011): 694-707.

[98]. Y Chen, Y.K Wang, K. Ugur, M.M. Hannuksela, J. Lainema, and M. Gabbouj, "The Emergence of MVC Standard for 3D Video Services," *EURASIP Journal on Advances in Singal Processing*, Vol. 2009, Article ID 786015

[99]. Wang, Ye-Kui, Miska M. Hannuksela, Stéphane Pateux, Alexandros Eleftheriadis, and Stephan Wenger. "System and transport interface of SVC."*Circuits and Systems for Video Technology, IEEE Transactions on* 17, no. 9 (2007): 1149-1163.

[100]. Sirannon, available online at http://sirannon.atlantis.ugent.be/

[101]. Su, Yeping, Anthony Vetro, and Aljoscha Smolic. "Common test conditions for multiview video coding." *JVT-T207*, Klagenfurt, Austria (2006).

[102]. Liu, Zhao, Yuansong Qiao, Brian Lee, Enda Fallon, Karunakar AK, Chunrong Zhang, and Shuaijun Zhang. "Experimental Evaluation of H. 264/Multiview Video Coding over IP Networks." *ISSC, Trinity College Dublin* (2011).

[103]. Micallef, Brian W., Carl J. Debono, and Reuben A. Farrugia. "Error Concealment Techniques for H. 264/MVC Encoded Sequences." In *IEEE Proc. of Int. Conf. of Electrotechnical and Computer Science (ERK), Portoroz, Slovenia*. 2010.

[104]. Anthony Vetro, Morgan McGuire, Wojciech Matusik, Alexander Behrens Jinho Lee, Hanspeter Pfister, "Multiview Video Test Sequences from MERL," *ISO/IEC JTC1/SC29/WG11 m12077*, Busan, Korea, April 2005.

[105]. Martinian, Emin, Alexander Behrens, Jun Xin, and Anthony Vetro. "View synthesis for multiview video compression." In *Picture Coding Symposium*, vol. 37, pp. 38-39. 2006.

[106]. Flierl, Markus, Aditya Mavlankar, and Bernd Girod. "Motion and disparity compensated coding for multiview video." *Circuits and Systems for Video Technology, IEEE Transactions on* 17, no. 11 (2007): 1474-1484.

[107]. Chung, Tae-Young, Il-Lyong Jung, Kwanwoong Song, and Chang-Su Kim. "Virtual View Interpolation and Prediction Structure for Full Parallax Multi-view Video." In *Advances in Multimedia Information Processing-PCM 2009*, pp. 543-550. Springer Berlin Heidelberg, 2009.

[108]. Hu, Jinhui, Ruimin Hu, Zhongyuan Wang, Mang Duan, Rui Zhong, and Zhen Han. "Least square based view synthesis prediction for multi-view video coding." In *Advances in Multimedia Information Processing–PCM 2012*, pp. 241-250. Springer Berlin Heidelberg, 2012.

[109]. Morvan, Yannick, and Dirk Farin. "Platelet-based coding of depth maps for the transmission of multiview images." In *Electronic Imaging 2006*, pp. 60550K-60550K. International Society for Optics and Photonics, 2006.

[110]. Lowe, David G. "Object recognition from local scale-invariant features." In*Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, vol. 2, pp. 1150-1157. IEEE, 1999.

[111]. Fischler, Martin A., and Robert C. Bolles. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography." *Communications of the ACM* 24, no. 6 (1981): 381-395.

[112]. Big O notion, lecture notes, MIT, Available online at http://web.mit.edu/16.070/www/lecture/big_o.pdf

Appendices

Appendix A

MVC Software Manual

Version: JMVC 8.5 (CVS tag: JMVC_8_5)
Last update: March 26, 2011

The JMVC (Joint Multiview Video Coding) software is the reference software for the Multiview 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). Since the MVC project is still under development, the JMVC Software as is also under development and changes frequently.

The JMVC software is written in C++ and is provided as source code. Section 0 describes how the JMVC software can be obtained via a CVS server. Information about the structure of the CVS repository is presented in section **Error! Reference source not found.**. Section **Error! Reference source not found.** describes how the JMVC software can be build on Win32 and Linux platforms, and section **Error! Reference source not found.** gives basic information about the binaries that are contained in the JMVC software package.

Accessing the latest JMVC Software

In order to keep track of the changes in software development and to always provide an up-to-date version of the JMVC software, a CVS server for the JMVC software has been set up at the Rheinisch-Westfälische Technische Hochschule (RWTH) Aachen. The CVS server can be accessed using WinCVS or any other CVS client. The server is

configured to allow read access only using the parameters specified in Table 0.1. Write access to the JMVC software server is restricted to the JMVC software coordinators group.

Table 0.1: CVS access parameters

authentication:	pserver
host address:	garcon.ient.rwth-aachen.de
path:	/cvs/jvt
user name:	jvtuser
password:	jvt.Amd.2
module name:	jmvc

Example 1 shows how the JMVC software can be accessed by using a command line CVS client.

Example 1: Accessing the JMVC software with a command line CVS client

```
cvs -d :pserver:jvtuser:jvt.Amd.2@garcon.ient.rwth-aachen.de:/cvs/jvt login
cvs -d :pserver:jvtuser@garcon.ient.rwth-aachen.de:/cvs/jvt checkout jmvc
```

In Example 2, it is shown how a specific JMVC software version – specified by a tag (JMVC_2_1 in Example 2) – can be obtained using a command line CVS client. Note that *co* represents an abbreviation for the command *checkout*, which was used in Example 1.

Example 2: Accessing the JMVC software version with the tag JMVC_2_1 with a command line CVS client

```
cvs -d :pserver:jvtuser:jvt.Amd.2@garcon.ient.rwth-aachen.de:/cvs/jvt login
cvs -d :pserver:jvtuser@garcon.ient.rwth-aachen.de:/cvs/jvt co –r JMVC_2_1 jmvc
```

Example of Configuration Files Used

```
# JMVM Configuration File in MVC mode

#====================GENERAL==============================

InputFile                D:\MVC\ballroom\ballroom
```

```
OutputFile              D:\MVC\ballroom\ballroom_QP31_4GOP3v

ReconFile               D:\MVC\ballroom\rec_QP31_4GOP3v

MotionFile            D:\MVC\ballroom\motion_QP31_4GOP3v

SourceWidth            640       # input  frame width

SourceHeight           480       # input  frame height

FrameRate              25.0      # frame rate [Hz]

FramesToBeEncoded      250       # number of frames

#====================CODING================================

SymbolMode             1         # 0=CAVLC, 1=CABAC

FRExt                  1         # 8x8 transform (0:off, 1:on)

BasisQP                31        # Quantization parameters

#====================STRUCTURE=============================

GOPSize                4     # GOP Size (at maximum frame rate)

IntraPeriod            4     # Anchor Period

NumberReferenceFrames  3         # Number of reference pictures

InterPredPicsFirst     1         # 1 Inter Pics; 0 Inter-view
```

```
Log2MaxFrameNum         11        # specifies max. value for frame_num (4..16)

Log2MaxPocLsb           7         # specifies coding of POC's (4..15)

DeltaLayer0Quant        0         # differential QP for layer 0

DeltaLayer1Quant        3         # differential QP for layer 1

DeltaLayer2Quant        4         # differential QP for layer 2

DeltaLayer3Quant        5         # differential QP for layer 3

DeltaLayer4Quant        6         # differential QP for layer 4

DeltaLayer5Quant        7         # differential QP for layer 5

MaxRefIdxActiveBL0      2         # active entries in ref list 0 for B slices

MaxRefIdxActiveBL1      2         # active entries in ref list 1 for B slices

MaxRefIdxActiveP        1         # active entries in ref list for P slices

#============================MOTION SEARCH===============

SearchMode              4         # Search mode (0:BlockSearch, 4:FastSearch)

SearchFuncFullPel       3         # Search function full pel

                        #   (0:SAD, 1:SSE, 2:HADAMARD, 3:SAD-YUV)

SearchFuncSubPel        2         # Search function sub pel
```

```
                    #   (0:SAD, 1:SSE, 2:HADAMARD)
SearchRange             96        # Search range (Full Pel)
BiPredIter              4         # Max iterations for bi-pred search
IterSearchRange         8         # Search range for iterations (0: normal)
#============================LOOP FILTER==================
LoopFilterDisable       0         # Loop filter idc (0: on, 1: off, 2:
                    #   on except for slice boundaries)
LoopFilterAlphaC0Offset 0         # AlphaOffset(-6..+6): valid range
LoopFilterBetaOffset    0         # BetaOffset (-6..+6): valid range
#============================WEIGHTED PREDICTION==========
WeightedPrediction      0         # Weighting IP Slice (0:disable, 1:enable)
WeightedBiprediction    0         # Weighting B  Slice (0:disable, 1:explicit,
                                  2:implicit)
#================== PARALLEL DECODING INFORMATION SEI Message
PDISEIMessage           0         # PDI SEI message enable (0: disable, 1:enable)
PDIInitialDelayAnc      2         # PDI initial delay for anchor pictures
```

```
PDIInitialDelayNonAnc   2          # PDI initial delay for non-anchor pictures



#=========================sEQUENCE PARAMETER SET

NumViewsMinusOne 2          # (Number of view to be coded minus 1)

ViewOrder              0-2-1   # (Order in which view_ids are coded)

View_ID                        0          # (view_id of a view 0 - 1024)

Fwd_NumAnchorRefs           0          # (number of list_0 references for anchor)

Bwd_NumAnchorRefs           0          # (number of list 1 references for anchor)

Fwd_NumNonAnchorRefs    0          # (number of list 1 references for non-anchor)

Bwd_NumNonAnchorRefs    0          # (number of list 1 references for non-anchor)

View_ID                1

Fwd_NumAnchorRefs           1

Bwd_NumAnchorRefs           1

Fwd_NumNonAnchorRefs     1

Bwd_NumNonAnchorRefs     1

Fwd_AnchorRefs                 0 0
```

Bwd_AnchorRefs 0 2

Fwd_NonAnchorRefs 0 0

Bwd_NonAnchorRefs 0 2

View_ID 2

Fwd_NumAnchorRefs 1

Bwd_NumAnchorRefs 0

Fwd_NumNonAnchorRefs 0

Bwd_NumNonAnchorRefs 0

Fwd_AnchorRefs 0 0

Appendix B

HTTP Packet Variant Transmitter Application – Server

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Threading;
namespace ServerApp
{
    public partial class FrmMain : Form
    {

        private DataLayer.Server Starter;

        public FrmMain()
        {
            InitializeComponent();
            Starter = new DataLayer.Server();


        private void CmdRun_Click(object sender, EventArgs e)
        {

            LblServer.Text = "Configuring MVC Server Application...";
LblServer.Refresh();
            Thread.Sleep(300);

            Starter.Configure();
            Starter.RegisterTCPChannel();

            LblServer.Text = "Starting MVC Server Application...";
LblServer.Refresh();
            Thread.Sleep(300);
            if (Starter.Run())
            {
                CmdRun.Text = "Started";
                LblServer.Text = "The MVC Server is started";
                Starter.StartListening();
                CmdRun.Enabled = false;

            else
                LblServer.Text = "The Server could not started.  please check
your security settings";
```

```
        private void CmdStartStop_Click(object sender, EventArgs e)
        {


            if (CmdStartStop.Text == "Stop Listenning")
            {
                Starter.StopListening();
                CmdStartStop.Text = "Start Listenning";
                LblServer.Text = "The MVC Server is stopped";


            else
            {
                Starter.StartListening();
                CmdStartStop.Text = "Stop Listenning";
                LblServer.Text = "The MVC Server is started";






        private void cmdadd_Click(object sender, EventArgs e)
        {
            OpenFileDialog of = new OpenFileDialog();

            if (of.ShowDialog() == System.Windows.Forms.DialogResult.OK)
                lstVideos.Items.Add(of.FileName);

            of.Dispose();

            UpdateVideoList();




        private void cmdremove_Click(object sender, EventArgs e)
        {
            if (lstVideos.SelectedIndex != -1)
            {
                lstVideos.Items.RemoveAt(lstVideos.SelectedIndex);
                UpdateVideoList();



        private void UpdateVideoList()
        {
            try
            {
                if (System.IO.File.Exists("c:\\tmp.txt"))
                    System.IO.File.Delete("c:\\tmp.txt");

                using (System.IO.StreamWriter sw = new
System.IO.StreamWriter("c:\\tmp.txt"))
                {
                    // Add some text to the file.
```

```
                    string strmsg = "";
                    for (int i = 0; i < lstVideos.Items.Count; i++)
                    {
                        if (i == 0)
                            strmsg = lstVideos.Items[i].ToString();
                        else
                            strmsg = strmsg + "\r\n" +
lstVideos.Items[i].ToString();



                    sw.Write(strmsg);

                    sw.Flush();
                    sw.Close();
                    sw.Dispose();



            catch { ;
```

HTTP Packet Variant Transmitter Application – Client

```
using System.Windows.Forms;

namespace ClientApp
{
    public partial class FrmClient : Form
    {
        public FrmClient()
        {
            InitializeComponent();


        private void button2_Click(object sender, EventArgs e)
        {
            Application.Exit();


        private void CmdPing_Click(object sender, EventArgs e)
        {
            LblProgress.Text = "Connecting to MVC Server...";
LblProgress.Refresh();


Utilities.Utility.setServerIP(Properties.Settings.Default.serverIPandPort);
            Utilities.Utility.RegisterTCPChannel();
            if
(Utilities.Utility.PingDataAccessTier(Int32.Parse(Properties.Settings.Default.po
cketsize)))//Make Sure It's connected
            {

                LblProgress.Text = "Connected to MVC Server";
```

```
                LblProgress.Refresh();
                CmdPing.Text = "Connected";
                CmdPing.Enabled = false;

            else
                LblProgress.Text = "Couldn't connected to MVC Server";



        private void button1_Click(object sender, EventArgs e)
        {



        private void cmdadd_Click(object sender, EventArgs e)
        {
            String[] videolist = Utilities.Utility.GetVideoList();
            lstVideos.Items.Clear();
            foreach (string a in videolist)
            {
                lstVideos.Items.Add(a);



        private void cmdRetreive_Click(object sender, EventArgs e)
        {
            System.IO.FileInfo a = new
System.IO.FileInfo(lstVideos.Items[lstVideos.SelectedIndex].ToString());
            string file = a.Name;
            double startsend;
            startsend = DateTime.Now.TimeOfDay.TotalMilliseconds;
            Utilities.Utility.ReadVideo("c:\\" + file ,
lstVideos.Items[lstVideos.SelectedIndex].ToString());

            double diff2 = DateTime.Now.TimeOfDay.TotalMilliseconds - startsend;
            MessageBox.Show(diff2.ToString() + " Milliseconds");

            lstdownloaded.Items.Add("c:\\" + file);
```

Appendix C

Sirannon Application Manual

SIRANNON 1.0.0

MODULAR MULTIMEDIA STREAMING

Alexis Rombaut

Introduction

Sirannon is a flexible and modular media server, client and proxy. It distinguishes itself by providing a modularity that manifests itself in how the user controls and configures the streamer. Each configuration describes graph of components, each handling basic video operations such as reading frames, packetizing frames and transmitting packets. The streamer handles both video and audio, something left out in many experimental streamers. Finally, the program supports variety of protocols such as RTP, RTSP, RTMP and HTTP. Jump to chapter installation if you immediately want to start installing the sirannon. The latest stable version is always available on http://sirannon.atlantis.ugent.be. The latest addition is a brand new GUI featuring graph drawing, XML editing and console execution for multiple configurations using tabs.

Streaming, Modular, Multimedia, Open source

Fact Sheet

Supported Codecs

- MPEG1 Video & Audio
- MPEG2 Video & Audio
- MPEG4 Video & Audio
- H264 AVC & SVC
- VP6, VP8/WEBM
- Vorbis
- AC3
- AMR-NB, AMR-WB

Supported Containers

Note, even if the sirannon supports a container, it still needs to support the codecs within the container.

- AVI
- MOV/MP4/F4V
- FLV
- WEBM
- MPEG2 (MPEG2 Program Streams)
- TS (MPEG2 Transport Streams)
- RAW (containing any of the supported codecs)

Supported Protocols

Sirannon can be both server and client for the following protocols:

- RTSP/RTP/UDP
- RTMP/TCP
- RTMPT/HTTP
- HTTP
- RTP/UDP
- UDP
- TCP
- Apple Live HTTP Streaming

Media Server

Universal Server - Universal Client - Protocol translation

The strongest feature is the combination of universal server (RTSP, HTTP, RTMP, RTMPT) and universal client (RTSP, HTTP, RTMPT, RTMPT). This combination gives Sirannnon the ability to transcode one protocol to another in real-time, dynamically and for many users. For example a request of the form

rtmp://mysirannon.com/RTSP-proxy/www.tv-world.net/content/AJaXo93cdW.mov in a Flash Player will make it connect to a Sirannon server that will in its turn connect to the fictional site www.tv-world.net using RTSP, request the stream and in real-time

change to protocol and packetization to sent it to the client using RTMP. The following table provides the supported protocol translations.

To -- From RTMP, RTMPT, HTTP, RTSP

Connecting to the media server

You can run Sirannon as media server (HTTP, RTMP, RTMPT, RTSP) by placing content in the folder "dat/media" and running:

sirannon dat/xml/media-server-std.xml dat/media

All requests to the Sirannon media server are of the form:

url ::= <protocol> "://" <server> "/" <application> "/" <request>

request ::= <server> "/" <application> "/" <request> |

When using the application FILE or HTTP, the request is a path to a file:

rtmp://sirannon.atlantis.ugent.be/FILE/flash/example.flv

http://sirannon.atlantis.ugent.be/HTTP/mp4/example.mp4

When using the Sirannon server as proxy for another protocol the applications are: RTMP-proxy, RTMPT-proxy, RTSP-proxy and HTTP-proxy. In this case request contains the server, application and file to request:

http://localhost/RTMP-
proxy/vod01.netdna.com/play/vod/demo.flowplayer/metacafe.flv

Media Server URL Examples

HTTP

http://" <server> "/" <HTTP|FILE@CONTAINER>

http://sirannon.atlantis.ugent.be/HTTP/demo.mov

http://sirannon.atlantis.ugent.be/FILE@FLV/mysequence.mkv

Apple Live HTTP

The short form

http://" <server> "/APPLE/"

http://sirannon.atlantis.ugent.be/APPLE/demo.mov

The long form

http://" <server> "/M3U/" <server> "/FILE@TS/"

http://sirannon.atlantis.ugent.be/M3U/sirannon.atlantis.ugent.be/FILE@TS/demo.mov

RTMP

"rtmp://" <server> "/FILE/"

rtmp://sirannon.atlantis.ugent.be/FILE/mysequence.mov

RTMPT

"rtmpt://" <server> "/FILE/"

rtmpt://sirannon.atlantis.ugent.be/FILE/mysequence.mov

RTSP

"rtsp://" <server> "/FILE/"

rtsp://sirannon.atlantis.ugent.be/FILE/mysequence.mov

Installation

Refer to the README in the distribution for the documentation about the compiling sirannon.

Tutorial

Introduction

This chapter describes how to construct, using the user interface, a basic streaming solution. Chapter describes several examples of streaming solutions without directly specifying the construction in the user interface. The following will create a basic

streamer for streaming a trailer from Apple using RTP. The output from the user interface is an XML configuration file. The next chapter describes how to run the sirannon with this configuration file.

Exploring the user interface

Launch the user interface by running:

sirannon.py

The user interface will launch and present itself in five tabs: Construct, Draw, XML, Run and Library. This tutorial will focus on the Draw and Run tab. Initially the draw area is empty and should look like in figure fig:6. Many of the functions available in the menu bar such as save, quit, undo, \textellipsis are self-explanatory.

Creating your first component

In order to open the trailer, a reader component is needed. The ffmpeg-reader component provides access to the Quicktime container. Let us create this component now. Right-click anywhere in the work area or click on the button 'New' in the toolbar. A menu will appear with the different categories of components. Select reader, ffmpeg-reader. A new component will appear as seen in fig:7. If you left click inside the new component, in the right of the screen an overview of the parameters of the component appears. These parameters are set at sensible defaults, we only need to fill out the filename of the trailer. After filling out the filename, the result should look as in figure fig:8.

Creating your second component

The component ffmpeg-reader creates packets containing video and audio frames. However, such frames are too large to be sent directly on the network: they have to be packetized into smaller packets. Now let us create the second component. Right-click in the empty work area and select packetizers, PES-packetizer in the menu. Change the name of the component to PES-packetizer-video by clicking on the button next to "name" in the properties area in the right of the screen. Drag this new component to a fitting place, as shown in figure fig:9.

Connecting your components

The two components need to be connected with each other. When you left-drag from inside one of the blue squares of a component, you will start drawing a line. Left-drag from inside the component ffmeg-reader and release the mouse inside one of the blue squares of the component avc-packetizer. The connection between the two components is now made and the configuration should now look as in figure fig:10.

Creating and connecting your third component

The PES-packetizer-video component will only process video frames. We need a similar packetizer for audio frames. Let us create a third component PES-packetizer-audio, found as packetizer, PES-packetizer in the menu. Change the name of the component to PES-packetizer-audio. Connect ffmpeg-reader with PES-packetizer-audio. No parameters have to be changed for this component either. The result should be as in figure fig:11. How can the component ffmpeg-reader know on which

connection to send packets, since video frames should be sent to AVC-packetizer and audio frames to MP4-packetizer? In the sirannon each packet is tagged with a number called xroute. If you look at the parameters from ffmpeg-reader, notice the parameters video-route and audio-route. Using the default settings, video packets will be tagged with xroute 100 and audio packets with xroute 200. Left click anywhere on connection between ffmpeg-reader and AVC-packetizer. In the properties area in right of the screen the parameter xroute appears. Since we want to sent only video packets over connection, fill out the value 100. By default the xroute is 0, meaning all packets are accepted on this connection. If a packet can take multiple paths, for example if multiple connections share the same xroute value, a separate copy will be sent over each of those connections. Now left-click on the connection between ffmpeg-reader and MP4-packetizer and fill out the value 200, causing audio packets to be sent over this connection.

Creating a transport stream

If we want to send the trailer over a single connection we need to multiplex the trailer into an MPEG2 Transport Stream. Create new component TS-multiplexer by selecting multiplexer, TS-multiplexer from the menu. Connect both packetizers with this new component. The result should look in figure fig:99. If you made a mistake, you can always undo using CTRL-Z or selecting undo in the menu.

Scheduling the packets

We need a scheduler to add real-time behavior: it stores packets in a buffer and releases them at the correct time. Create a scheduler by selecting schedulers, frame-scheduler

from the menu. Figure fig:12 shows the intended result. If your draw area becomes too small you can always zoom out by clicking the appropriate button in the toolbar.

Transmitting the packets

The frames are ready now for transmission: they are packetized in sufficiently small packets, multiplexed into an MPEG2 transport stream and scheduled at the correct time. Let us create an RTP-transmitter. Select transmitter, RTP-transmitter from the component menu and connect the scheduler with the new transmitter. We have to fill out the source port and destination address. In the properties area for the parameter port fill out the value 5000 and for the parameter client fill out $127.0.0.1:1234$. Also set the parameter debug to true for this component. Figure fig:13 shows the result.

In order to close the sirannon after streaming the sequence, we should place a special block called sink at the end of the chain. After the last packet of the sequence has passed through this sink, it terminates the program gracefully. Select system, sink from the component menu and make a connection form the transmitter to this sink. The result should look as in figure fig:14. Zoom out or use best fit if the drawing area is too small.

Saving and Executing your the configuration

Now that the configuration is complete, we can save it. In the menu bar under File use either Save or Save as or press CTRL-S. We can now run sirannon using the tab "Run". A new tab should appear that looks like in figure fig:98.

If the GUI finds the binary it should appear as first item of the command line options. Under Unix you should have compiled and installed the sirannon binary if you wish to

continue. If you installed the binary in a different location you can always select that location.

We do not need to fill out the configuration, when you press play the GUI will automatically using the current configuration unless you overwrite it.

Press the large Play button. The console window expands and shows the output of the sirannon process. If you filled out an incorrect container for the ffmpeg-reader, sirannon should fail with an error like Unhandled RuntimeError: core.ffmpeg-reader: Could not open file(iTrailer.mov). If you do not have a sample video at hand you can always download the demo containers from http://sirannon.atlantis.ugent.be/files/demo.tar.bz2. Bear in mind that for MPEG2 Transport Streams only MPEG video and audio codecs are supported. If you created everything correctly output should appear in the console as in figure fig:97.

Once the console is running you can open for example VLC Media Player. Under Media, select Open Network Stream. Fill out rtp://@:1234 as URL. VLC should now start playing the video. \textbf{CAVEAT: If your sequence uses H.264/AVC, VLC will not have critical information that was transmitted by streamer before VLC launched. You will need to press Stop followed by Play to fix it. To prevent this behaviour you can set the parameter repeat-parameter-sets to true for the component ffmpeg-reader.

Now the video should be playing as in figure fig:96. If it is not working, go over the following check list.

- Press Stop followed by Play
- Is parameter destination for RTP-transmitter set to 127.0.0.1:1234?
- Does the console show an error such as Could not open file or Unsupported codec?

Internal view

We shortly describe here the internal view of the sirannon in order to understand the parameters in the next section. The basic operation of the sirannon is single threaded. The execution consists of a series of cycles with each cycle aiming to process one frame. The real duration of such a cycle, for example 2ms, is often much lower than the duration of one frame, typically 40ms. The process can sleep during the difference, lowering the CPU load considerably.

Execution parameters

Let us introduce three execution parameters. To modify these parameters in the user interface for a configuration, go in the menu bar to Settings, Settings. These parameters are specific for each configuration.

quantum: in milliseconds, defines the minimum time to process one frame. If processing of a frame took less than this quantum, the process sleeps during the difference, lowering the CPU load. If you match this quantum with the duration of each frame, you get a working performance with minimal CPU load. For example a stream

with 25 frames per second or 40 ms per frame, can be streamed using less than 1\% CPU using a quantum of 40ms. If the quantum is set at 0, the process never sleeps, producing very accurate timing (order 1 ms) at the cost of 100\% CPU utilization. default: 0

simulation: in microseconds, if this value is greater than 0 the sirannon runs in simulated time. Each cycle the simulated time is increased with this value. This has the advantage of providing arbitrary time precision and possibly faster execution at the cost of losing the real time behavior. For example the simulation of streaming an HD/H264 stream using simulation steps of $40000\mu s$, can be run in $2s$ instead of $40s$ in real time. When simulation is defined, the parameter quantum is ignored. default: 0

seed: if this value is greater than 0, it provides the seed for the random numbers in the sirannon. If the value is 0, the current time is used as seed. default: 0

Command line parameters

In order to use the configuration files without having to edit them when using differnt content, component parameters (not execution parameters!) can be entered in the form of \$1, \$2, \$3\textellipsis The symbol \$n will be replaced by the n^{th command line parameter after the configuration file (see next section). Use the symbol \$\$ to circumvent this interpretation and represent the character "\$".

Usage

verbatim

sirannon [-cvbh] [-q=NUM[ns|us|ms|s]] [-s=NUM[ns|us|ms|s]] [-r=NUM] FILE [ARG-1] ... [ARG-n]

Run the program with FILE as configuration.

Options:

-h Help information

-b Build information

-c Overview of components

-v Verbose, use up to 4 v's to increase the level

-q=NUM Quantum in milliseconds

-s=NUM Simulation in milliseconds

-r=NUM Seed for the random number generator

Arguments:

ARG-1 Replace any occurrence of "$1" in FILE with ARG-1

ARG-2 Replace any occurrence of "$2" in FILE with ARG-2

ARG-NUM Replace any occurrence of "$NUM" in FILE with ARG-NUM\end{verbatim

Example 1

sirannon -v -q=5 avc-streamer.xml gladiator.avi 127.0.0.1:5000

In this example we run the configuration file "avc-streamer.xml" with two command line parameters: the video file name and destination address. These command line parameters will replace the symbols \$1 and \$2 entered in the configuration file. In addition we set the quantum to 5 ms and print debug messages from toggled components.

Example 2

sirannon -b -c

Prints the build information and the available components. Since no xml file is specified, the program end instantly.

Examples

The basics of the sirannon were explained in the previous chapters. By using examples, we will demonstrate the many possibilities of the sirannon. In contrast with the tutorial, we will not explain how to construct the configuration in the user interface but we will focus instead on describing the function and structure of different configurations. Sometimes we will refer to specific parameters of components. For a detailed description of the components and a full list of available parameters, refer to the next

chapter. The distribution of Sirannon contains for each example the corresponding XML file.

Example 1: a basic streamer

Let us start with a basic example, even simpler than the tutorial: a basic AVC streamer. Figure fig:ex:1 shows the schematic. It contains the four basic components: a reader, a packetizer, a scheduler and a transmitter, connected in a chain.

Example 2: a basic receiver

The stream sent by the basic streamer can be received by a media player, but the sirannon also can act as a receiver. Hence, the sirannon is more than just a streamer. Figure fig:ex:3 shows this receiver. The receiver has four components like the basic streamer, performing the reverse operation: a receiver, a scheduler, an unpacketizer and a writer. Received packets are unpacketized into the original frames and those are written back to a file. The scheduler avoid problems such as UDP reordering or duplication. However, the RTP protocol should circumvent these problems, making the scheduler superfluous in this example.

Example 3: differentiated streaming

This example modifies the structure of the basic streamer from example 1 in order to stream over multiple connections. Figure fig:ex:2 shows the result. Instead of having one rtp-transmitter, we now have three. We also add a frame-classifier to differentiate the I, P and B frames. The sirannon allows forks in the schematic using a simple routing mechanism based on a label xroute per packet. The reader gives each packet an

initial xroute of 100. A classifier increases the xroute\ with a fixed value for each classification. In this example, frame-classifier has three parameters I, P and B with values 1, 2 and 3 respectively, producing packets with xroutes 101, 102 and 103. The connections between frame-classifier and rtp-transmitter 1, 2 and 3 use xroutes 101, 102 and 103 respectively to split the stream. Section sec:route also explains this routing system.

Example 4: proxy

A differentiated stream cannot be played by standard players such as VLC or Quicktime, because it only accepts one connection for video, not three for example. The sirannon can function as a proxy, converting a differentiated stream into a single stream. Such stream can then be received by standard players. Figure fig:ex:4 shows the schematic. The function consists of three types of components: three receivers, a scheduler and a transmitter. The packets from the three rtp-receivers pass through the scheduler. In this setting, the scheduler is anything but superfluous since the three connections are unsynchronized. The scheduler restores the original order. It uses additional information included by the transmitters in the RTP header extension since the sequence numbers and time stamps from the RTP connections are insufficient to restore the original order. It also gives the merged stream the correct time behavior, so that the rtp-transmitter can send the merged stream. This merged stream is then received by a standard player.

Example 5: a packet loss generator

The sirannon can also run as an offline tool. Figure fig:ex:5 shows the schematic to introduce packet loss, with a different percentage for each type of frame. The frames are read and packetized by avc-reader and avc-packetizer. The frame-classifier splits the stream into I, P and B packets. For each of these types there is different component random-classifier 1, 2 or 3 with its specific packet loss. The damage streams are merged and unpacketized by avc-unpacketizer and the resulting frames (some of the original frames are lost) are written by the component writer.

Example 6: using transport streams

The sirannon supports MPEG2 transport streams, widely used in digital television. It multiplexes video, audio and meta-data into one stream that we can send over one connection, as opposed to the default RTP mode where each video, audio or meta-data substream has its own RTP session. Figure fig:ex:6 shows the configuration. It does not differ that much from the basic streamer. ffmpeg-reader opens a Quicktime file containing both video and audio frames and xroute is set at 100 and 200 respectively for video and audio by the reader. Two packetizers create generic PES packets for video and audio respectively. The component ts-mutliplexer multiplexes these packets into one transport stream. The two remaining components schedule and transmit the transport stream packets.

Example 7: using and constructing blocks

Let us create a simulator to test the behavior of a buffer. Block components allows us to group several component into one component, allowing more readable and parameterized configurations. Figure fig:ex:7a shows a qmatch-buffer component and

three block components. Each block components uses a configuration file that reads and packetizes a sequence. Figure fig:ex:7b shows this configuration. This looks similar to the basic streamer example but it has at the end of the chain an out component which sends packets to the surrounding configuration (figure fig:ex:7a). In order to obtains precise timing results, we run the sirannon in simulation mode with a simulation step of 1000 μs.

Example 8: massive simulation

Using blocks we can create massive simulators of for example 18 different streams, as shown in figure fig:ex:8.

Extending sirannon

In the source tree src/Misc/Example.cpp describes a good commented example about writing your own component. Place any new sources you create in the folder src/Local and add the flag --enable-local to configure. When adding new sources, rerun configure --enable-local.

Components

Components can have two special parameters:

debug: if true, the component prints debug information, requires verbose level one or higher (use -v in the command line), default: false

thread: if true, run the component in a separate thread, default: false

Appendix D

Virtual View Stitching

```
vassar_1 = imread('D:\0-Thesis\Chapter 6\Frames\ballroom\0_1.bmp');
vassar_3 = imread('D:\0-Thesis\Chapter 6\Frames\ballroom\2_1.bmp');
newimage = vassar_1(:,21:end,:);
half3 = vassar_3(:,end-40:end-21,:);
image2 = uint8(zeros(480,640,3));
%imshow(r);figure(gcf);
image2(:,1:620,:)= newimage(:,:,:);
%imshow(r);figure(gcf);
image2(:,621:640,:)= half3(:,:,:);
imwrite(image2,'D:\0-Thesis\Chapter 6\Frames\ballroom\v_1.bmp')
imshow(image2);figure(gcf);
```

Disparity Compensated Predicted Virtual View

```
%running the programming to extract the middle view from the two
%main program
Image1= 'D:\0-Thesis\Chapter 6\Frames\ballroom\0_1.bmp';
Image3= 'D:\0-Thesis\Chapter 6\Frames\ballroom\2_1.bmp';

windowSize = 55;
DisparityRange = 79;
[disparity, Image2,time,dis13]=
reconstructionNEW(Image1,Image3,windowSize,DisparityRange);

figure(2);imshow(Image2);
imwrite(Image2,'D:\0-Thesis\Chapter 6\Frames\ballroom\vDisp_1.bmp')

image_or2=imread('D:\0-Thesis\Chapter 6\Frames\ballroom\1_1.bmp');
PSNR_value=PSNR(im2double(rgb2gray(image_or2)),im2double(rgb2gray(Imag
e2)));

%view reconstruction functions


function [disparity, Image2, time,dispmap]=
reconstructionNEW(Image1,Image3,WindowSize,DisparityRange)

Image1=imread(Image1);%read image 1
Image3=imread(Image3);%read image 2
tic; %start the clock
[dispmap costmap,pcost,wcost] = desparitymatch( Image1, Image3,
WindowSize, DisparityRange, 0); %retun disparity map.
%we are passing values. the 0 means no subpixel disparity. change to 1
for
%sub pixel disparity. the values returned
time=1000*toc; %stop clock


disparity=dispmap;
```

```
disparity=floor(disparity/2);

halfImage1=floor(size(Image1,2)/2);


Image1H=Image1(:,1:halfImage1);

for i=1:size(Image1H,1)
    for j=1:size(Image1H,2)
        Image2(i,j,:)=Image1(i,j+disparity(i,j),:);


    end
end

for i=1:size(Image1H,1)
    for j=1:size(Image1H,2)
        Image2(i,j+halfImage1,:)=Image3(i,halfImage1+(j-
disparity(i,j+halfImage1)),:);


    end
end
```

Panoramic Virtual View Rendering

```
im1 = imread('ballroom_1.bmp');
im2 = imread('ballroom_3.bmp');
image_or2=imread('ballroom_2.bmp');


fshowH=figure;
subH1=subplot(2,2,1); imshow(im1);
subH2=subplot(2,2,2); imshow(im2);
movegui(fshowH, 'northwest');

%% sift features
figure(fshowA); figure(fshowH);
[pts1 pts2] = SIFTmatch( im1, im2, 5, true );


%% ransac homography
figure(fshowA);
[im2_TH, best_ptsH] = ransac( pts2, pts1, 'proj_svd', 10 );
showbestpts(subH2, subH1, best_ptsH);
figure(fshowH);

%% stitch homography
[im_stitchedH, stitched_maskH, im1TH, im2TH] = stitch(im1, im2,
im2_TH);
figure(fshowH);
subplot(2,2,3); imshow(im1TH);
subplot(2,2,4); imshow(im2TH);
```

```
fH=figure;
axis on;
movegui(fH, 'northeast');
imshow(im_stitchedH);

image2H = im_stitchedH(1:480,15:654,:);
imwrite(image2H,'ballroom_2_pan.bmp');

PSNR_value=PSNR(im2double(rgb2gray(image_or2)),im2double(rgb2gray(imag
e2H)));
```

Virtual View created using Sub-pixel Disparity Prediction

Virtual View created using L0 Smoothing before Disparity Prediction