A PORTABLE STEREO-VIDEO STREAMING SYSTEM

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

EMIN ZERMAN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

SEPTEMBER 2013

Approval of the thesis:

A PORTABLE STEREO-VIDEO STREAMING SYSTEM

submitted by **EMIN ZERMAN** in partial fulfillment of the requirements for the degree of **Master of Science in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Gönül Turhan Sayan
Head of Department, **Electrical and Electronics Engineering** _____

Prof. Dr. Gözde Bozdağı Akar
Supervisor, **Electrical and Electronics Engineering** _____

Examining Committee Members:

Prof. Dr. A. Aydın Alatan
Electrical and Electronics Engineering Department, METU _____

Prof. Dr. Gözde Bozdağı Akar
Electrical and Electronics Engineering Department, METU _____

Assoc. Prof. Dr. Cüneyt F. Bazlamaçcı
Electrical and Electronics Engineering Department, METU _____

Assoc. Prof. Dr. Ece Güran Schmidt
Electrical and Electronics Engineering Department, METU _____

Dr. Cevahir Çığla
Aselsan _____

Date: **September 04, 2013**

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: EMIN ZERMAN

Signature :

ABSTRACT

A PORTABLE STEREO-VIDEO STREAMING SYSTEM

Zerman, Emin

M.S., Department of Electrical and Electronics Engineering

Supervisor : Prof. Dr. Gözde Bozdağı Akar

September 2013, 90 pages

In the last decade, 3D technologies have made a great advancement in reaching the end-users. Many of the cutting-edge technology had the opportunity to reach the customers. With the increase in popularity of mobile electronics and the new mobile 3D multimedia displaying methods, the mobile 3D became a very important multimedia factor in the last decade. This thesis presents an implementation of real-time stereo-video capture, compression, and wireless streaming from embedded platforms to mobile devices, as an end-to-end system. There are different aspects of an end-to-end system. In order to create an efficient and fast video streaming system, we tested different encoding structures for H.264/AVC and H.264/MVC with different 3D video formats. Experiments on different video streaming techniques are conducted, and the most effective one is employed. On the final system, side-by-side Frame Packing Arrangement presentation of stereo-video is used in H.264/AVC encoding system. The system is realized by using DM3730 SoC ARM device as transmitter and a mobile device with autostereoscopic display as receiver. Presented architecture makes 3D video communication between the mobile users connected to a wireless network possible. Performance tests are conducted for the system, and the results are presented in this study.

Keywords: Mobile 3D video, Real-time 3D video streaming, Embedded systems, Mobile devices, End-to-end system

ÖZ

TAŞINABİLİR BİR STEREO-VİDEO AKTARIM SİSTEMİ

Zerman, Emin

Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü

Tez Yöneticisi : Prof. Dr. Gözde Bozdağı Akar

Eylül 2013 , 90 sayfa

Son on yıl içerisinde 3B teknolojileri son kullanıcıya ulaşma konusunda oldukça ilerlemiştir. Yeni gezgin 3B gösterim yöntemleri ve de artan toplum beğenisi ile birlikte son on yıl içinde gezgin 3B çok önemli bir çokluortam öğesi olmuştur. Bu tez bir uçtan uca sistem olarak gerçek-zamanlı stereo-video yakalama, sıkıştırma ve gömülü platformlardan gezgin aygıtlara kablosuz duraksız iletim gerçeklemesini sunmaktadır. Verimli ve hızlı bir video aktarım sistemi oluşturabilmek için farklı 3B gösterim biçimleri ile H.264/AVC ve H.264/MVC için farklı kodlama yapıları denenmiştir. Farklı video aktarım yöntemleri üzerinde denemeler yapılmış ve en etkili olanı seçilmiştir. Sistemin son halinde H.264/AVC kodlama sistemi içerisinde yan-yana stereo-video gösterimi kullanılmıştır. Sistem, gönderici olarak DM 3730 SoC ARM işlemcili bir aygıt ve alıcı olarak da otostereoskopik ekranlı bir gezgin aygıt kullanılarak gerçeklenmiştir. Sunulan mimari, kablosuz ağa bağlı olan gezgin kullanıcılar arasında 3B video iletişimini mümkün kılmaktadır. Sistem için başarım testleri gerçekleştirilmiş ve sonuçlar bu çalışma içerisinde sağlanmaktadır.

Anahtar Kelimeler: Gezgin 3B video, Gerçek-zamanlı 3B duraksız video aktarımı, Gömülü sistemler, Gezgin cihazlar, Uçtan-uca sistem

To My Family

ACKNOWLEDGEMENTS

First of all, I would like to thank my thesis supervisor Prof. Dr. Gözde Bozdağı Akar for her patience, wisdom, understanding and guidance. I am grateful for her helps through my studies and introducing me to the Multimedia and Computer Vision areas. It has always been a pleasure to work with her.

I would like to state that this thesis is a result of an efficient multitasking. Multimedia Research Group helped a lot to this multitasking. I would like to thank Prof. Dr. A. Aydın Alatan for his technical insight, Yağız Aksoy for his companionship and his helps in this thesis work, Yeti Ziya Gürbüz for his friendship and the frank personality, Emrecan Batı for his caring character and technical helps, Beril Beşbınar and Ozan Şener for their friendship and making the laboratory a fun place to be. I am also thankful for the advices and helps of Döne Buğdaycı. I consider myself lucky for being in the same environment with Ömürcan Kumtepe, Erhan Gündoğdu, O. Serdar Gedik, Ahmet Saracoğlu, Dr. Engin Tola, and Dr. Emrah Taşlı. I also acknowledge Ahmet Orkun Tomruk for his helps on the hardware issues.

I would like to thank Türk Telekom Arge for providing the opportunity to work in a state-of-the-art project. I would also thank The Scientific and Technological Research Council of Turkey (TÜBİTAK) for their financial support during my M.Sc. studies.

Thanks to my cousins Cem Vedat Işık, Dr. Işıl Işık Gülsaç and Ass. Prof. Dr. Lokman Onur Uyanık. They have always been helpful through my college life as well as being a role model for my life and my academic career.

Last but not least, I would like to express my gratitude for my family. They have always been there for me. I would like to thank my father Hüseyin Zerman for his wisdom and patience, my mother Necibe Zerman for her love and kindness, my aunt Adviye Zerman for her care and support, my brother Oğuz Zerman for his existence and ceaseless energy . I also want to thank Harika Başpınar, who helped and encouraged me through from the beginning, for her love.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

1.1 Motivation

Entertainment and art have always been a source of curiosity for humanity. Through centuries, the search for a new form of artistic expression method has never ended. Multimedia technologies carry this flag in today's world. The demands of the people increase with the ease in access to cutting-edge technologies and advancing technology. Multimedia engineers and researchers try to handle the problems in the process of presenting state-of-the-art technologies to people to use in their daily lives.

The 3D multimedia is not a new concept. The first 3D creation concept is stereo-vision which is also quite popular today. Stereo-vision is used firstly in 1838 with a mirror-device by Sir Charles Wheatstone [23]. The first stereoscopic TV was proposed in 1920s. The first boom of the 3D in movie theatres was in 1950s in the United States of America. The movie creators and the theatre owners tried to hold the viewers by introducing 3D movies as a reaction to the television which is getting popular [59, 23, 25, 2]. The second boom of 3D in movie theatres happened in 2000s. The release of blockbusters in 3D format supported this trend. In 5 years, from 2006 to 2010, the number of worldwide 3D screens are increased from 258 to 21,936 [9]. Hence, 3D researches are important for the movie theaters.

After the invention of television, 3D technologies are started to be introduced in television systems also. Even though there are different works on 3D multimedia delivery on the TV systems, the television systems generally rely on stereoscopic 3D representation systems to remain on the safe side with respect to the back-compatibility. The 3D TV researches on the Europe started with the 3DTV project which is funded by 6^{th} Framework of EU. The general works done on 3DTV improved the 3D researches on Television Systems. These works are continued on the Mobile 3DTV project which is also funded by 7^{th} Framework of EU. This project generated end-user products such as an auto-stereoscopic handset communicating over DVB-H result of Mobile 3DTV project [29].

Given the state of the 3DTV researches, both researchers and industry tries to reach the mobile device users because the mobile technologies are getting popular, and more and more people start to use mobile phones, tablets, and smart devices. The smartphone usage in the US rose from 36 % to 56 % in the last 2 years [70]. The usage statistics among smart phone users show that most of the users utilize the Internet for social media and multimedia purposes [27]. With this increasing profile of smart-phone and internet users, the demand for mobile 3D multimedia is expected to increase.

Figure 1.1: System Overview

There are different works about 3D multimedia delivery to mobile devices. These mainly use the Digital Video Broadcasting-Handheld (DVB-H) technology [2, 4, 10, 30], which uses the VHF-III and UHF IV electromagnetic wave band intervals [21]. There are some studies about streaming 3D multimedia to mobile devices over the Internet; however, these works are not real-time and most of them requires reconstruction, an additional computational cost on the receiver side [68, 40].

The mobile systems using DVB-H as their physical medium works in the real-time. Though, these systems require DVB-H transmitters with large power requirements. The use of the Internet as the physical medium could solve both the coverage and power problems. In order to reach mobile customers, wireless internet connections such as 3G or 4G could be used. With the increasing bandwidth, the 3G and 4G technologies, or namely EDGE, WCDMA, HSPA, LTE, are more and more favored by the customers [36]. This advancement in mobile wireless data links permits huge data transfers among the mobile users. This is an advantage against the DVB-H.

On the real-time aspect, the streaming applications which runs over DVB-H are mostly real-time. However, there are some studies about the transmission of 3D content to the other side over the Internet that does not ensure real-time performance [40, 68]. Nonetheless, these systems lack the support of mobile systems and they require specialized equipment such as glasses.

In this thesis, the main motivation is to develop a low power system containing end-to-end chain of devices and software that streams the stereo-video over the Internet for the portability and accessibility issues.

1.2 Scope of the Thesis

This thesis proposes an end-to-end system that utilizes mobile or embedded platforms on both end in order to keep the system movable and low power which uses the wireless technologies as its backbone rather than the other methods such as DVB-H. An embedded platform with ARM architecture have been used in order to keep the proposed system due to the advantage of ARM as being low power, hence mobile.

The proposed end-to-end system includes all the following components starting from capture to display as it is shown in Figure 1.1. These parts can be listed as follows:

- Capture or creation of the content
  - Stereo capture of real world by utilizing a Linux library for video processing
- Video compression

  - Experimentation and comparison among various video compression techniques for a 3D video encoding system

- Streaming
  - Analysis and comparison of different video streaming and transmission protocols without the consideration of a special case for 3D video

- Rendering 3D content and Display
  - Creation of a live stereo media player by the modification of a well known open source software: VLC Media Player for Android

1.3 Outline of the Thesis

In order to explain the proposed system in a better way, firstly some of the concepts related to 3D multimedia should be explained. In Chapter 2, the human 3D perception is discussed at first. Then, different techniques for 3D display are revised. For a better coverage of both the 3D concepts and the 3D image/video representation methods are given.

The basic principles of video compression and different 3D video compression techniques are explained in detail in Chapter 3. Two different experiments are conducted on both desktop computer and embedded platform with respectively Joint Multiview Video Coding (JMVC) Reference Software 8.5 and x264 Encoder for different purposes such as comparison among different encoding methods and the comparison of the encoding times with respect to the encoding structure on transmitter platform. Experimental work and the results relevant to the 3D image representation methods and 3D video compression is given in this chapter as well.

In Chapter 4, the video streaming topic is discussed. First, a survey of the existing methods for video streaming is done. Then, the methods for video streaming over the Internet are analyzed in detail. In desired end-to-end mobile system, the most important properties in terms of network are the flexibility of the stream packets to different network structures and the adaptivity to these different network structures. The analytical comparison related to the video streaming considering these issues is also presented here.

In Chapter 5, the proposed system is explained. Before starting, the related work is presented and discussed on the difference among all similar systems. After a brief system overview, the transmitter side and the receiver side are explained in detail. This chapter covers all the system parts from one end to the other. Both the integration and other modifications made are represented in this chapter.

In Chapter 6, the conclusions from the thesis and a brief summary are presented. The future works and the application areas are indicated in the last chapter.

CHAPTER 2

3D MULTIMEDIA TECHNOLOGIES

The purpose of research is to find a new way of looking and solutions to existing problems, to make a discovery or invention that will ease the lives of the mankind. Every research topic has an ultimate goal. In 3D multimedia technologies, this ultimate goal is to find a way to represent the view as it is seen in the real world from all points of view, namely free-view television or display. Even though there are some examples with both hardware [84], and software (with 3D reconstruction) [64], the Free Viewpoint TVs (FTV) are not widely used and are not appropriate to advance into the commercial area in both display technologies and content creation.

3D multimedia technologies are a cornerstone in the way to reach the free-view television. The developments in the 3D multimedia paves the way in both displaying technologies and the content creation aspects. Even though 3D TVs are becoming widespread in the last few years, very large portion of the newly created 3D content originates from 2D content. For these reasons, 3D multimedia is still a hot topic for research.

In order to understand the 3D multimedia technologies, we should first understand the human 3D perception and different techniques about 3D content formats and compression. The following sections discuss these issues.

2.1 Human 3D Perception

Human visual system (HVS) consists of eyes and brain. The actions of eyes and brain can be considered as two separate parts of a system [95, 29]. Eyes are used for reception of the light or scattered light rays from the outside world, and act as sensors. The brain, on the other hand, is an organ which understands and recognizes the sensor inputs. Perception is a whole different concept from seeing the outside world. Perception includes many different psychological phenomena, and affected by the previous experiments of the viewer and environmental conditions. Perception may also be affected by the disabilities of the eyes with different illness conditions; however, this is not in our scope. In order to analyze and choose an appropriate display for 3D purposes, we must first understand how people perceive 3D world and depth feeling.

There are different aspects in depth or 3D perception that depend on both the sensory part, eyes, and the recognition part, brain. The components or the factors of depth perception cannot be separated as sensory and recognition. There are different studies on 3D percep-

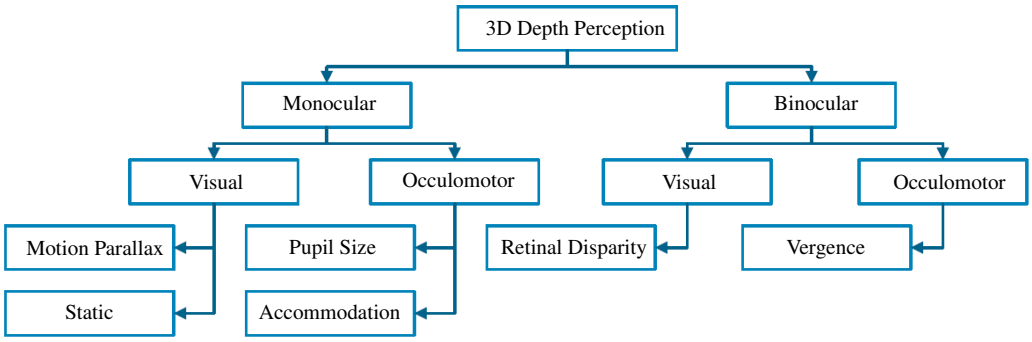


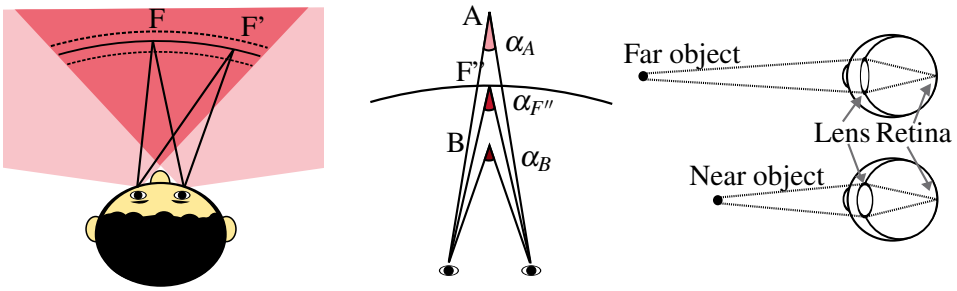
Figure 2.1: Factors of 3D Depth Perception in Human Visual System, [63]

tion modalities and these works suggest that monocular and binocular depth perception is independent from each other [28]. As the depth is perceived independently by monocular and binocular systems, we can analyze first the monocular part and then the binocular part.

The affecting factors of depth perception can be separated as monocular and binocular as shown in Figure 2.1. The monocular cues are divided into two parts as visual and occulomotor (muscular feedbacks from eyes). Monocular visual depth cues are mostly related to previous experiences and they can be extracted from 2D images [31, 28, 63, 11]. These depth cues can be listed as follows:

- **Interposition:** Position differences and occlusions between different objects in the scene shows relative depth of the objects.
- **Linear Perspective:** Explicitly seen edge or any linear perspective source leads to the perception of depth in terms of the given regular subject sizes or learned perspective feeling.
- **Relative and Known Size:** Comparison of known sizes of two objects gives the feeling of 3D depth as their relative sizes differ. Two of the same object with different sizes implies that the object that appears smaller is farther away.
- **Texture Gradient:** Texture of same size and shapes shows whether there is any distortion on that plane as a linear perspective or not.
- **Light and Shadow Distribution:** Light and shadow distribution in the scene gives different clues about the depth of the objects. Just lighting and shadow appearance of an object carries information on its 3D location.
- **Aerial Perspective:** Different artifacts such as blurriness, dust, humidity or fog can imply depth information. For example, in natural landscape images, the farther the scene object gets, the more blurry the image becomes.

Beside the static visual clues, the other most important monocular visual depth cue is the motion parallax. This is the effect which is created by either the movement of the environment or the observer's head. Rate of movement on perpendicular 2D plane gives the depth information in motion parallax. If the movement of the object is greater for the same motion



(a) Visual Area and Horopter (b) Angular Disparity (c) Accommodation

Figure 2.2: Binocular and Monocular Occulomotor Depth Perception Cues

of the observer, the object is close to the observer. If the movement rate is very low or zero, then the object or plane is far away from the observer.

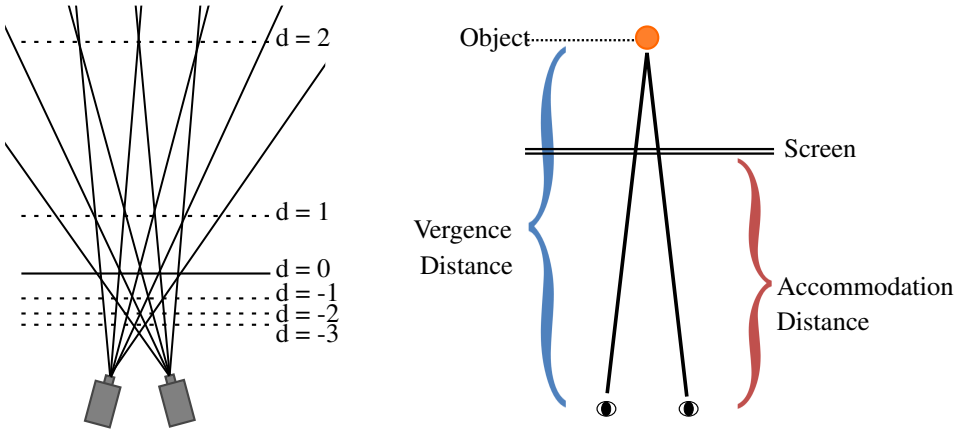
Apart from the monocular visual depth cues, occulomotor depth cues also have importance in depth perception. These depth cues are pupil size and accommodation. The pupil size controls the amount of incoming light. The amount of light rays and the optical properties of the incoming light rays change the obtained image reflection on the retina. Smaller pupil size leads to a smaller hole for light and this increase the *Depth of Focus*, the depth interval in which all the objects can be seen as single focused or nearly focused rather than crossed or uncrossed images. Also, the pupil size arranges other effects such as diffraction and aberration.

Another monocular occulomotor depth cue is the accommodation of the lenses in eyes. The accommodation is defined as the process of getting a clear focused image by changing the diameter and thickness of the lens. The feedback sent from the muscles changing lens to the brain helps humans to find the depth of the object focused. An example to the difference in lens dimensions within the accommodation process can be seen in Figure 2.2 (c). Accommodation is a very helpful depth cue for human depth perception system.

As indicated above, HVS can perceive depth with even using only one eye by utilizing the monocular visual and occulomotor depth cues. However, binocular depth cues are more precise and help HVS in different situations such as the relative depth estimation, breaking camouflage, perception of surface material and surface curvature judgement [31].

The binocular depth cues are also divided into two parts as occulomotor and visual. Binocular occulomotor depth cue is the vergence. The vergence is defined as the movement of two eyeballs in different directions [11]. The convergence of two eyes on a point on the area of the intersection of the two eyes as shown in Figure 2.2 (a) is needed to perceive the object in 3D. The point of convergence creates a point cloud in space that all the points are observed as single with the same convergence and the accommodation of the eyeballs. This area is called Panum's fusional area [31] and the center-line of this area is called horopter.

The stereoscopic difference of the depth is created by the angular disparity, the difference values between the horopter and the point of the object of interest. The angular disparity is recognized by the brain when a different object is in front or behind of the focused point. The angular disparity between points B and F", shown in Figure 2.2 (b), can be calculated as:



(a) Effect of Disparity

(b) Accommodation and Vergence

Figure 2.3: Effects of Stereo Pixel Disparity on Depth Perception. The effect of stereo disparity on perceived depth (a) and vergence-accommodation rivalry (b).

$$\delta = \alpha_{F''} - \alpha_B \tag{2.1}$$

According to [31] and [63], minimum noticeable depth distance is between 0.3 and 0.84 mm, found after the trigonometric calculations of the distance between two eyeballs and the screen counting on the stereo acuity as 10" and 20" (seconds of an arc) respectively. Their works take the average screen distance as 650-670 mm and an average human eye separation as 65 mm. According to the calculations the depth perception decrease with the increasing screen distance [11]. This also shows that the binocular depth cue of vergence works particularly on closer distances.

Another binocular visual depth cue is the retinal disparity between the stereo images projected on the retina of two eyeballs. The disparity is calculated after the two eyeballs are focused on the same location on an object. This retinal disparity phenomena is used in the stereoscopic display systems also. Sign of disparity show the side as in front or behind of the screen. The disparity of two retinal images is calculated as it is shown in angular disparity equation above. A sample image of disparity on the screen between the stereo image pair and its effect is shown in Figure 2.3 (a).

One of the most effective 3D depth cues is the fusion of the vergence and the accommodation cues. However, with a screen 3D display device, this fusion could not be realized. Generating the content by utilizing the retinal disparity depth cue is the most common way. However, the vergence and accommodation does not hold the same place in these display types and this situation disturbs the observers. The vergence is made on the object placed in front of the screen. However, the accommodation must be remained on the screen in order to see the screen in focus. This situation is represented in Figure 2.3 (b) and this issue is an important topic for research on displays [66, 12].

According to [56, 58], the most important depth cue among all of the ones discussed above are the binocular stereo image disparity and the motion parallax. Thus, binocular vision is an important factor in human 3D depth perception. The effects and the importance of binocular

vision in 3D display systems are discussed in the following section.

2.2 Techniques for 3D Display

3D display of the generated content in a natural way is the most desired goal of the 3D display designers. However, the physical limitations do not let that conditions to come true. Most of the 3D display systems do not supply a natural view condition for 3D content. All or most of the above depth cues must be satisfied in order to achieve a natural 3D display system.

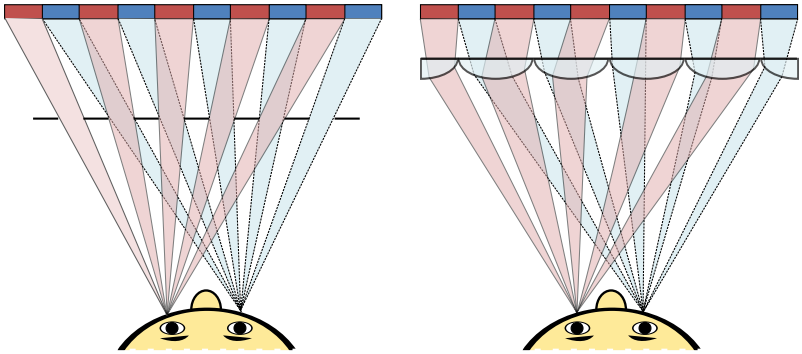
There are different 3D display systems. Most of them utilizes the idea of stereoscopy. Stereoscopy means solid seeing ($\sigma\tau\varepsilon\rho\varepsilon o\varsigma$ - *solid* and $\sigma\kappa o\pi\varepsilon\omega$ - *to look*) as Fehn stated [23]. In order to see the objects in their solid form, the stereoscopic view is required due to different depth cues explained in previous section. In stereoscopy, two different eyeballs acquire two different images. In order to make this operation possible, different displaying methods can be used. The main division is done as the displaying methods by the main principle of creation of 3D depth sense such as binocular, holographic, and volumetric.

2.2.1 Binocular Displays

As the first way of utilizing the stereo view, binocular displays are created since the first emergence of the 3D technology on great scale. This method lets different images to be shown to different eyes. In order to send two different images to two different eyes, mankind used mirrors and glasses as the first tools. The first binocular display was the Wheatstone mirror stereoscope [13, 23]. This device uses two mirrors for two eyes in 45° form. Another early example of binocular display is the Brewster stereoscope which was a special kind of glasses that the images are attached in front of the glasses [13].

Most binocular display methods use multiplexing methods for displaying. In these methods, mostly active or passive glasses are used [63, 59, 31]. In *wavelength-division* or *color multiplexing method*, stereo image pair is combined by overlaying onto each other by using different/non-overlapping wavelength colors. Examples of these colors are red-cyan, red-green, red-blue, amber-dark blue. Most of these color multiplexing methods causes to lose the color component of the 3D content. In *polarization-division multiplexing method*, two different polarized images, mostly generated by two projectors, are displayed on the same surface and the stereoscopic view is provided by polarized filtered glasses. In *time-division multiplexing method*, the images are shown with their original state and with doubled frame rate. The frames are shown in time multiplexed mode as even frames as left frame and odd frames as right. The active shutter glasses then is synchronized with the display in order to show the observer the right view to right eye by masking the wrong eye. All these displaying types are called *stereoscopic displays* and they require glasses for viewing.

The other type of the binocular displays is *autostereoscopic displays* which does not require any glasses to perceive 3D. These displaying systems use *space-division multiplexing* as their base displaying method. These systems have a specialized light manipulation layer between the screen such as parallax barriers and lenticular optics. The behavior of these systems are similar, and they intend to make sure the pixels of the right view are delivered



(a) Parallax Barrier (b) Lenticular Sheet

Figure 2.4: Two Different Autostereoscopic 3D Display Methods [19]

to right eye and pixels of left view are delivered to left eye, as shown in Figure 2.4.

Other than stereoscopic and autostereoscopic display systems, multiview displays and head mounted displays can also be classified as binocular display. Head mounted displays are very similar to the stereoscopic displays. Multiview displays are similar to autostereoscopic displays with more views. In order to increase the viewing angles and in order to make 3D display more realistic by utilizing motion parallax, multiview displays generate more than one view, up to 16 views [19], by using the video plus depth map representation. That decreases the discomfort caused by the lack of motion parallax. By multiview displaying technology, the 3D displays such as televisions become more usable and that may increase the public acceptance of 3D displays and televisions.

2.2.2 Holographic Displays

Holographic displays are similar to autostereoscopic displays in some aspects. They are generated by using stereoscopy principles and they are displayed onto a flat screen. However, holographic displays are using specialized materials in the display screen. These materials helps the light rays which are mostly generated by laser beams to diffract and delivered to right eye order. The advantages of the 3D holographic displays are being able to support binocular depth cues, motion parallax, accommodation issues [45]. Hence holographic displays are the most comfortable 3D displays for human observers.

2.2.3 Volumetric Displays

Volumetric displays are the display methods which uses a rotating mediums covering a certain volume and a projection method on that medium. Volumetric display of 3D content is better than any stereoscopic methods due to the natural look generated on the given volume. Some examples of the volumetric displays are rotating plane with a projector to display accurate 3D display [22, 79], a full parallax display with rotating mirror and the projection from the top of the system [37], and a cylindrical display for the delivery of the 3D

content [101]. Considered that the images are created on a volume, these systems simulates the real world and creates a solution for different problems such as motion parallax, accommodation-vergence rivalry, and full parallax display (by introducing vertical parallax). The disadvantage of these systems is the difficulty of 3D content generation for all angles.

These systems have different advantages and disadvantages for different areas of usage. The stereoscopic systems are most commonly used display methods due to easy implementation and back-compatibility. However, these systems require glasses which is an additional equipment that may bother some users. Autostereoscopic displays do not require those glasses; however, both stereoscopic and autostereoscopic displays can not support one of the most important depth cue; motion parallax. Autostereoscopic displays also suffers from vergence-accommodation rivalry. Holographic displays and volumetric displays offer real 3D sense. On the other hand, these displays are not suitable for mobile applications because of the requirements to special hardware that needs huge spaces.

2.3 3D Image and Video Representation Formats

In order to utilize the acquired 3D multimedia, a representation method is needed. There are different methods to represent 3D images on the given screens. The most basic method is the stereoscopy, as explained above. To use stereoscopy in 3D image representation, two views must be present. Different video representation formats require different video compression methods. Finding the most effective video compression method relies on the type of video representation format.

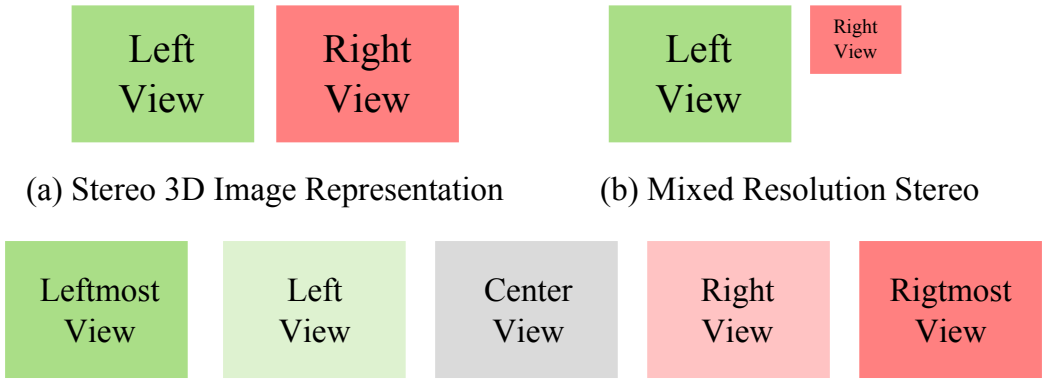
The difference in the representation formats can be grouped as the amount and quality of the sent frame size and any auxiliary data [72, 79, 71]. Most of the research community uses and tries to promote the stereo and multiview representation. On the other hand, the commercial community prefer to use the frame-compatible format due to the backward compatibility.

2.3.1 Stereo and Multiview Representation

In stereo and multiview representation, the views are separately captured and separately stored. The view number can start from 2 and go up to camera arrays of many cameras [93]. The basic idea about minimum two views is to simulate the two eyes of human. The left view is required for left eye and the right view is required for the right eye. More cameras than two are required to generate 3D view for different view generation processes. [3, 71] An example of stereo and multiview 3D representation can be seen in Figure 2.5.

The binocular suppression theory [76] states that the dominant high quality suppresses the lower perceived quality of the other eye. In the mixed resolution stereo which utilizes this information to reduce the bandwidth, the left view is left as full resolution and the right view is sub-sampled at half resolution as can be seen in Figure 2.5 (b) [54, 14].

This video representation presents a better quality and better 3D reconstruction because it is not subject to the quality degradations due to the downsampling, except mixed resolution stereo, which we will see in the frame-compatible representations section. Also, this video representation method, in multiview case, gives much more knowledge compared to the



(a) Stereo 3D Image Representation

(b) Mixed Resolution Stereo

(c) Multiview 3D Image Representation with 5 views

Figure 2.5: Stereo and Multiview 3D Image Representations

other methods, and that provides a better 3D estimation and reconstruction on the display side by reducing occlusion handling complexity [71].

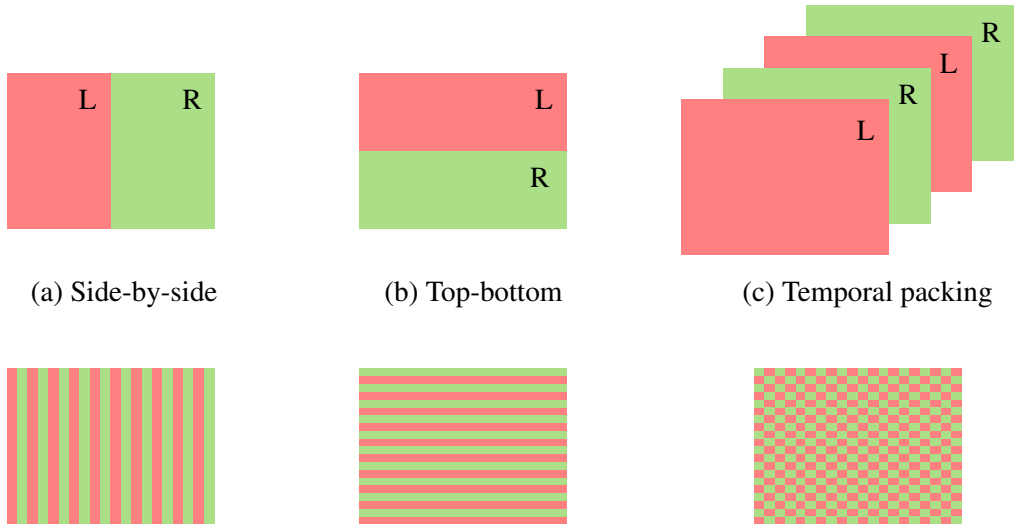
2.3.2 Frame-compatible 3D Representation

Another solution to the 3D image and video representation issue is to use a frame-compatible format. In this format, the frame sent is compatible with the frame size of the display medium. For that purpose, the frame-packing must be done. In order to fit the stereo content in a frame, different methods can be used such as spatial or temporal frame packing.

Spatial frame-packing methods include horizontal and vertical subsampling and interlacing. The stereo video frames can be stitched together after the subsampling or they can be interlaced horizontally or vertically. Examples of resulting frames can be seen in Figure 2.6 (a) and (b), respectively side-by-side and top-bottom. In Figure 2.6 (d) and (e), horizontal and vertical interlacing samples can be seen. However, these methods decrease the quality of the content as subsampling procedure introduces losses to the given frame. In order to decrease the losses on the interlacing process, the checkerboard interlacing can be done which an example of that method can be seen in Figure 2.6 [80].

Instead of using spatial frame-packing to reduce the errors due to the subsampling process, one can use temporal frame-packing method [91]. In this method, the left and right frames are interleaved in temporal dimension. In order to keep the frame size, one frame of left view and one frame of right view are sent one after another. An example of this method can be seen in Figure 2.6 (c). The advantage of this method is to keep the frame as original. However, as it is subsampled through the time, the upsampling process on the decoder side will be done on the time dimension. Hence, if the frame is dynamic, there will be some artifacts due to the upsampling process.

The frame-compatible 3D representation methods are preferred mostly by the commercial community as it is indicated before. The main reason of that is the frame-compatible methods are back-compatible. People with standard 2D televisions can watch a spatial or temporal frame-packed video by decoding only the left half or taking only left frames on the



(a) Side-by-side (b) Top-bottom (c) Temporal packing

(d) Vertical Interlacing (e) Horizontal Interlacing (f) Checkerboard Interlacing

Figure 2.6: Frame-compatible 3D Image and Video Representations

temporal dimension [93].

2.3.2.1 Color and Polarization Multiplexing

The color and polarization multiplexing are easier methods to use for both end-users and the producers. The color multiplexing (or anaglyph 3D) method is a very basic and easy way to create a frame-compatible representation. It is very easy for end users to find a color glasses and consume the generated content. However, the color multiplexing is subject to some errors such as loss of perception of color components and after effects such as color bleeding and nausea [38].

The polarization method is both simple and can reflect true colors, as it is discussed in the previous section. With the spatial or temporal interlacing methods, polarized views can be presented to the end-user.

2.3.3 Enhanced Video Streams

The other representation systems either require more bandwidth or introduce loss. In order to decrease the losses and save the bandwidth, enhanced video streams are used. As the enhancing part of these streams, mostly a depth information is used. Instead of transmitting stereo views, a video and a depth map can be transmitted and the secondary view can be rendered at the receiver side. This could save bandwidth and the 3D content would be transmitted to the recipient.

There are different methods to transmit the 3D content by sending the depth. Depth could be used for not only saving bandwidth but also improving the present system. These systems

utilize depth in different ways. In order to generate a 3D output with many viewing points, a dense depth map is very important [5].

Video plus Depth representation system, also known as 2D plus Z, uses the depth map to render the secondary view in order to generate the stereo image pair [89, 3]. The depth map is used to calculate the disparity magnitude and direction and rendering is completed by fusing the disparity knowledge with the base view received. However, occluded regions come into the new frame and holes appear in rendered image. Even though these holes can be filled by estimating the probabilities of the hole pixels and finding the nearest and most possible known patch [16, 48], quality of the rendered 3D pair will be affected.

The 2D plus depth with auxiliary views representation, or Layered Depth Video (LDV), format is used to avoid that problem [61, 71]. However, this representation format is applicable only to synthetic contents because the system needs the complete background image of the occluded area as auxiliary view.

The Multiview plus Depth (MVD) representation format is used to combine multiview video with multiple depth maps. Thus, the reconstruction or rendering of the 3D content on the receiver side will be easier to handle with both multiview video and multiview depth data [73, 41].

CHAPTER 3

3D VIDEO COMPRESSION

In order to transmit the captured video over the Internet, the video has to be compressed because of the bandwidth and speed limitations. This compression needs to be both computationally cheap and efficient. There are different methods for 3D video compression. The compression type depends on the 3D image or video representation format. This subject has been studied on different scales in the literature [92, 71, 10, 15]. In this section, these different works are discussed and the experimental setup is given by the result of the experiments conducted.

3.1 Methods for 3D Video Compression

3D compression methods differ by the 3D representation format. In order to compress 3D content, there are different methods used. Most of them are based on a special 3D image and video representation format. This section presents an overview to the commonly used 3D video compression methods.

The general opinion on the 3D video compression is to use the state-of-art compression technique H.264/AVC, or H.264/MPEG-4 AVC, proposed by ITU-T Video Coding Experts Group (VCEG) and ISO Moving Picture Experts Group (MPEG) communities jointly [20]. As it is explained thoroughly in the following subsections, majority of the 3D video coding research is based on H.264 [80].

There is a new video compression format named High Efficiency Video Coding (HEVC or H.265) proposed by the same team of ITU-T VCEG and ISO MPEG. Even though this video compression format was announced in April 2013 and published in June 2013, there is a study of 3D video coding on HEVC [55].

3.1.1 H.264/AVC Simulcast

For stereo multiview 3D image representation, the most basic method is to encode two, or more, different views as they are separate videos. In other words, the relevance and similarity between different views are not exploited [43, 50]. This encoding method is called simulcast because both streams, or views, are generated and transmitted at the same time, i.e. simultaneously as it is shown in Figure 3.1 (a).

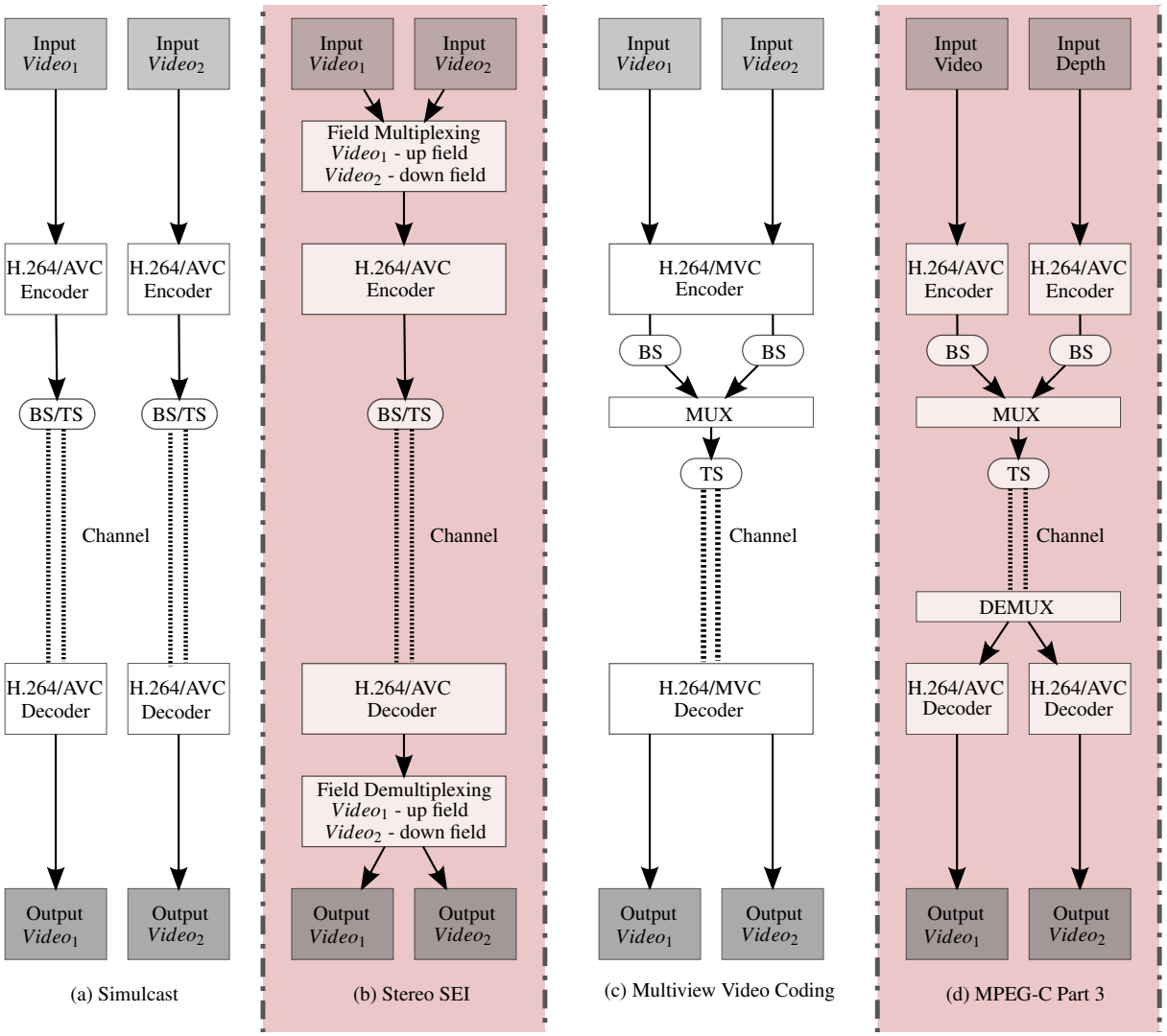


Figure 3.1: Methods for 3D Video Compression - Upper level shows the transmitter side and the lower level shows the receiver side

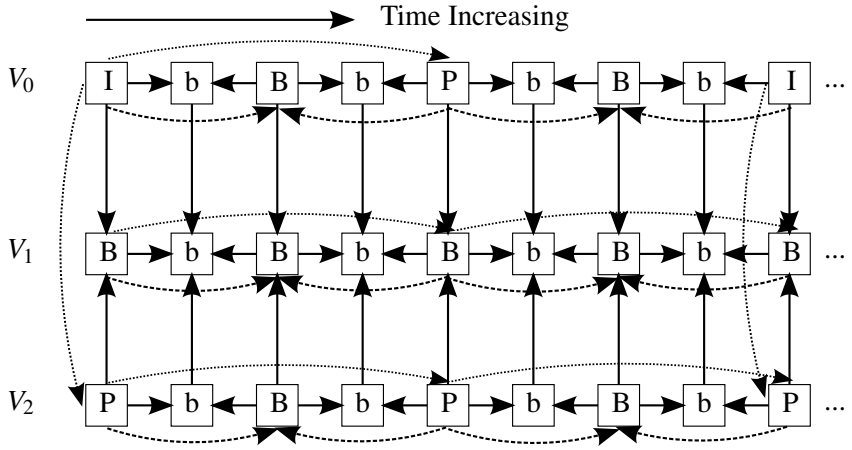


Figure 3.2: Multiview Video Coding Structure Showing the Inter-View Dependencies

The advantage of this encoding structure is its simpleness. It is easy to encode different views and transmit them at the same time, and the computational complexity is low on both transmitter and receiver side [86, 85]. The disadvantage of H.264/AVC Simulcast is the high bandwidth requirement compared to other methods.

3.1.2 H.264/AVC Stereo SEI Message

In this encoding method, the two views of stereo view representation is packed into a single frame by using the frame-compatible representation methods, such as side-by-side, top-bottom, checkerboard or line-interlaced, and encoding is carried out as a single video sequence as it is shown in Figure 3.1 (b). The stereo Supplemental Enhancement Information (SEI) message is set during the encoding progress according to the interlacing method in the pre-processing stage. In the initial state of the standard, the only accepted interlacing method was row interlacing; however, after an amendment published by ITU-T, all the interlaced formats discussed in Section 2.3.2 is accepted [50, 91]. This coding structure also enables the exploitation of inter-view redundancies [86].

The advantage of this method is the backward compatibility in both frame size and count of bitstreams acquired. However, there will be a quality reduction due to the spatial or temporal sub-sampling in the pre-processing stage.

3.1.3 Mixed Resolution Coding

In order to profit from the human 3D perception, the mixed resolution coding can be used. In mixed resolution representation, or coding, one of the stereo views -generally right view- is downsampled in order to save the bandwidth. The binocular suppression theory [76] states that the perceived quality depends on the dominant quality or higher resolution. Hence, using the mixed resolution representation, one can achieve a lower bitrate. However, the computational complexity on both transmitter and receiver increases in this coding structure with the downsampling and upsampling procedures [86, 85, 14, 43].

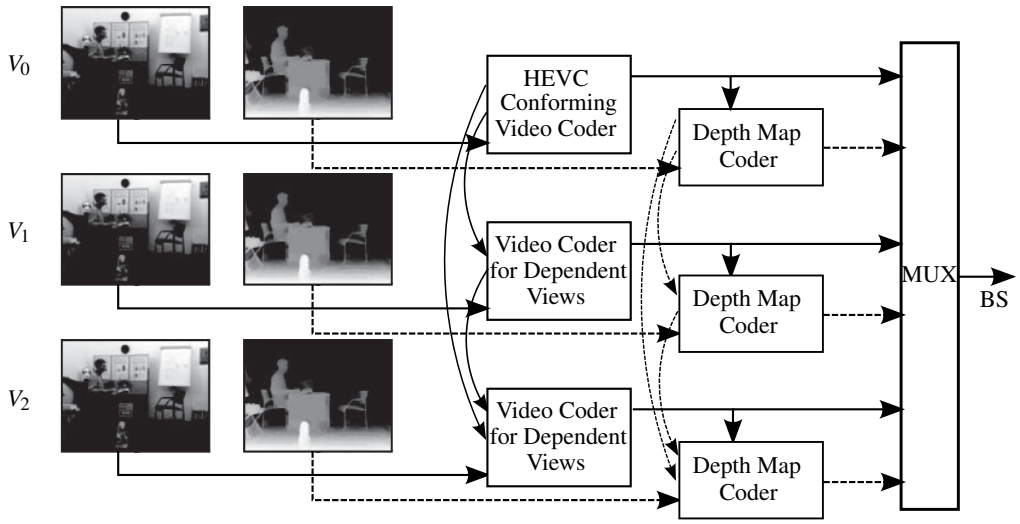


Figure 3.3: Multiview Depth Coding Structure

3.1.4 Multiview Video Coding

Multiview video coding is a specialized version of the H.264/AVC with an amendment and named as H.264/MVC. This encoding structure utilizes the similarities among the multiview, as well as stereo, videos and uses this knowledge to reduce the payload size of the encoded bitstream [43, 85, 50, 93]. An example of the MVC encoding structure is given in Figure 3.1 (c) and an example of utilization of redundancies among the views can be seen in Figure 3.2.

In this encoding structure, the views are encoded together and two separate bitstreams are generated. In MVC, two separate bistreams are generated in the first step of encoding even the redundancies between views are exploited. Those two bitstreams are multiplexed into one bitstream. This bitstream is transmitted to the receiver side and the H.264/MVC decoder is used to decode the acquired bitstream. Reduced payload size is an advantage; however, the computational complexity increases on the encoder side [54].

3.1.5 Video plus Depth Coding

Video plus depth (V+D) coding is a more efficient way to transmit 3D stereo video than H.264/AVC Simulcast method. The depth view has a less payload size than a second (or right) regular view. Also, different point of views can be rendered by using video plus depth representation. However, the quality of the 3D content reduces due to the artifacts caused by the rendering process.

Video plus depth coding is similar to the H.264/AVC Simulcast. The video and depth are encoded simultaneously and multiplexed into single bitstream. After transmitted, the bitstream is demultiplexed and the resulting two bitstreams are decoded separately. After decoding, a view rendering is required. This introduces a computational complexity on the receiver side [54]. An illustration of this process can be seen in Figure 3.1 (d).

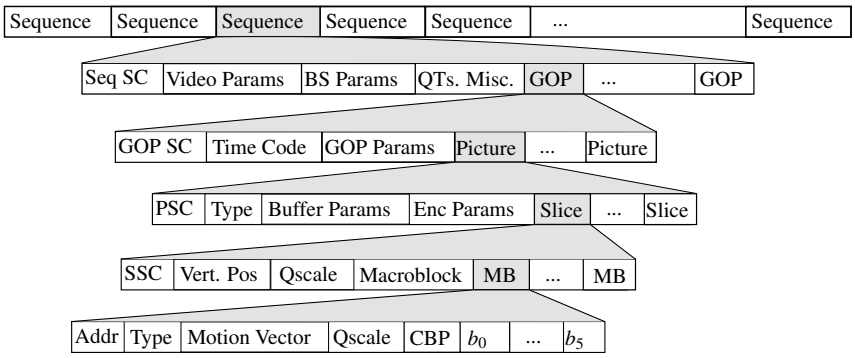


Figure 3.4: The MPEG Hierarchy Structure of Video Stream

3.1.6 Multiview Depth Coding

Multiview depth coding is an encoding structure that uses MVD representation type which is discussed in Section 2.3.3. This is an encoding method in which HEVC/H.265 video compression standard is used [42, 51]. This method is constructed as a union of both MVC and V+D coding methods. First, the views are coded by utilizing the redundancies among them while simultaneously their corresponding depth views are coded in the same structure. After the encoding process, these separate bitstreams are multiplexed into a single file and that file is transferred as shown in Figure 3.3.

3.2 Encoding Structures of H.264/AVC and H.264/MVC

In H.264/MPEG AVC encoding standard, there are different encoding structures. These encoding structures are important for the compression efficiency. As the main method of the video encoding is utilizing the redundancies between the blocks of a frame and between frames, different encoding structures offer better compression techniques by changing the prediction structure. In order to understand the encoding structures, the frame types in the a video compression system should be reviewed before.

There are three main different video frame coding types: Intra Coded Frame, Predicted Frame, and Bidirectionally Predicted Frame. These frames are generally placed in a Group of Picture (GOP) and these GOPs are placed in the MPEG transmission sequence as it is shown in Figure 3.4. There are other levels in this structure such as slices and macroblocks.

Macroblock is a 16x16 pixel sized the most small element of the video coding frame. The motion estimation and compensation tasks are done on the basis of the macroblocks. H.264/AVC employs different sizes in a 16x16 macroblock such as 8x16, 16x8, 8x8, 8x4, 4x8, and 4x4. Slices are composed of macroblocks and they are used to lessen the effect of network errors occurred during transmission [87]. Slices can be constructed by a number of macroblocks, and the number of macroblocks in a slice can differ from very little to whole frame. Frames, or pictures, are formed by slices, and the GOPs are formed by frames [96]. The differences among these frame types are explained below and a visual comparison can be seen in Figure 3.5.

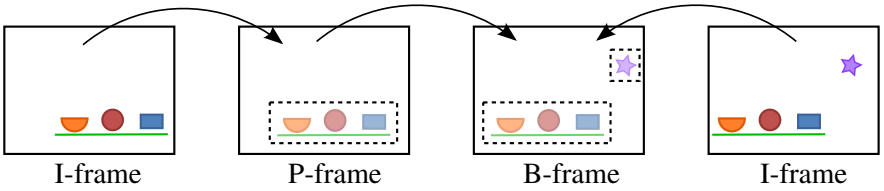


Figure 3.5: Different Frame Coding Types (Work of Petteri Aimonen, Public Domain)

Intra Coded Frame (I-Frame)

Intra Frame Coding is a frame coding type which does not need any external reference frame. This coding type is used to exploit the spatial in-frame redundancies such as the same color of background or any repetitive texture within the coded frame. However, in order to control the complexity, the search of redundancies within the same frame is limited by a parameter of the search range. Any redundancies are referenced in-frame and by that the total size of the frame is reduced. Instead of the reduction of the frame size, I-frame is the most larger sized frame amongst the I, P, and B frames. The first frame of a video sequence is always coded as an Intra Frame due to the lack of external reference frame.

Predicted Frame (P-Frame)

Predicted Frame Coding is a frame coding type which uses the older frame as external reference. This method exploits the temporal redundancies between two frames. The assumption of this idea is that the frames of video does not change very drastically in temporal dimension. By referencing most of the macroblocks from the older decoded frame, the size of a predicted frame reduces to a very small rate. The predicted frame can only take I and P frames as reference. So, the B frames cannot be a source of external reference to the coded P frame.

Bidirectionally Predicted Frame (B-Frame)

Bidirectionally Predicted Frame is very similar to P frame due to their referencing nature. B frame can utilize both older and newer (i.e. newer in temporal order of playing) frames as references. This gives the ability to compress the given video much more. However, the computational complexity increases with the increased search space. The advantage of the B frame is being B frame smaller size compared to the P frames in different circumstances such as moving texture or scene change in the sequence.

Group of Picture (GOP)

Group of Picture is a set of picture that is aimed to be decoded by using only the frames in the GOP. Hence, the GOP starts with an I frame and consists I, P, and B frames. This behavior of GOP permits various features such as fast forwarding, rewinding, and changing the playing location of video. When a playing location is changed, the nearest independent GOP is found by the media player software and the video is started to be decoded starting from the I frame of that GOP.

The GOP concept is created first on MPEG-1, and the JMVC reference software allows to crate GOPs without the I frames, instead GOPs with P frames are created when Intra Period

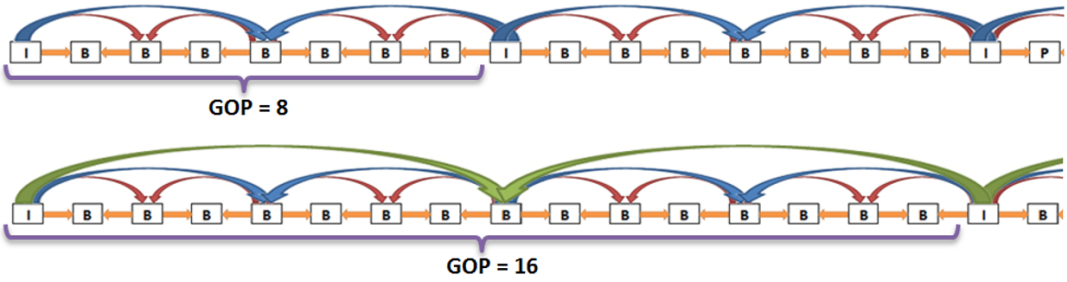


Figure 3.6: Different GOP Sizes in Hierarchical Encoding Structure

is greater than the GOP size. In that context, the GOP is used as a semi self-decodable unit due to the need to the I frame which is the reference of P frame to be decoded. The difference between the GOP size and the encoding structure can be seen in Figure 3.6 in which the GOP size and Intra Period are the same.

Intra Period

Intra Period is the frame period of encoding an I-frame. Intra period is different than the GOP size in H.264/AVC encoding standard. Intra period defines the refresh rate of an encoded video stream. The higher intra period becomes, the higher quality gets. Nonetheless, the video stream size gets larger in that case. The need for refreshing the stream and the bandwidth should be considered while setting this parameter. In the following experiments this parameter is set same as the GOP size.

Figure 3.7: Variation in the Image Quality by QP values of (Top-left) 16, (Top-right) 24, (Bottom-left) 36, and (Bottom-right) 48.

Quantization Parameter (QP)

Quantization Parameter is the parameter that is used in the quantization of the DCT coefficients in the encoding process. QP is not a parameter related to encoding structures. However, it is needed to be explained here because it is an encoding parameter and it is

changed in the experimental setup. The only lossy part of the encoding and decoding chain of the video aside network or transmission losses is the quantization process. This loss or degradation in the quality can be controlled by adjusting the QP. A visual example of the variation in image quality can be seen in Figure 3.7.

3.2.1 Encoding Structures on Single Stream (Simulcast)

The encoding structures in this section can be divided in two parts as the encoding structures on single stream and on multiview streams. The single stream part is as same as plain H.264/AVC encoding structures. Encoding and streaming multiple views or streams simultaneously is called simulcast. In the stereo case, either side-by-side, or video plus depth, or streaming two video separately can be considered as simulcast.

IPP

The IPP coding structure is simple and straightforward. The main operation is taking an I-frame in predefined intervals, and constructing P-frames in-between. The flow of operation follows a straight linear path, so the complexity of the decoding is not increased much. In encoding and decoding, encoder/decoder is only interested with the previous, or older, frame. In order to avoid accumulated errors on the video, I-frames are taken with a period conventionally as products of eight as shown in Figure 3.8.

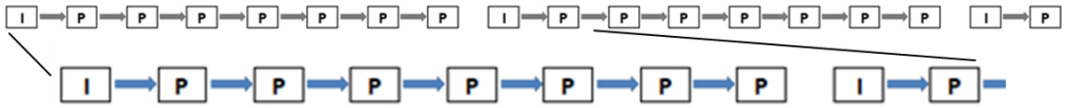


Figure 3.8: IPP Encoding Structure with GOP Size of 8

The advantage of the IPP encoding structure is its simpleness and the low complexity during the encoding. However, IPP encoding structure can propagate errors easily than Hierarchical encoding structure with large intra frame periods. That property may affect the visual quality perceived by the end user.

Hierarchical

The Hierarchical encoding structure utilizes B-frame encoding in order to get benefits of video compression by extending search space with 2 or more distinct frames. As explained earlier, B-frames can take both previous and next frames as external reference frames. For this operation, the next frame that the B-frame took as reference must be encoded before. In order to do that, the decoding order must be different than the playing order. Hence, this situation introduces a problem of buffering. In order to decode the hierarchical encoded stream, a buffer has to be kept for each arriving GOP. An example of hierarchical encoding structure with GOP size of 8 can be seen in Figure 3.9.

Figure 3.9: Hierarchical Encoding Structure with GOP Size of 8

The difference of the temporal frame order of the encoded video consisting of 10 frames and

the encoding, or decoding, order of the same video can be given as following:

Temporal Order :	0	1	2	3	4	5	6	7	8	9	10	frame #
	I	B_3	B_2	B_3	B_1	B_3	B_2	B_3	I	B_1	P	frame type
Decoding Order :	0	8	4	2	6	1	3	5	7	10	9	frame #
	I	I	B_1	B_2	B_2	B_3	B_3	B_3	B_3	P	B_1	frame type

The hierarchical encoding structure is much more effective in compressing when compared to the IPP encoding structure. However, the computational complexity increases due to the increase in the search space by 2. Also, the buffering need for GOP may constitute a problem when the GOP size increases.

3.2.2 Encoding Structures on Multiview Streams

In H.264/MVC encoding, multiview streams are used and the inter-view redundancy is exploited by referencing one of the views to the other. The main concepts of the encoding structures does not change but there are additions to them.

IPP Simplified

IPP Simplified encoding structure is used to simply utilize the redundancy between left and right views (in stereo case) by taking left view as a reference to the right view on the anchor frame, or I-frame, only. By doing so, the large size of the right frame's I-frame is reduced to a smaller value of a P-frame. An example of the IPP Simplified encoding structure can be seen in Figure 3.10

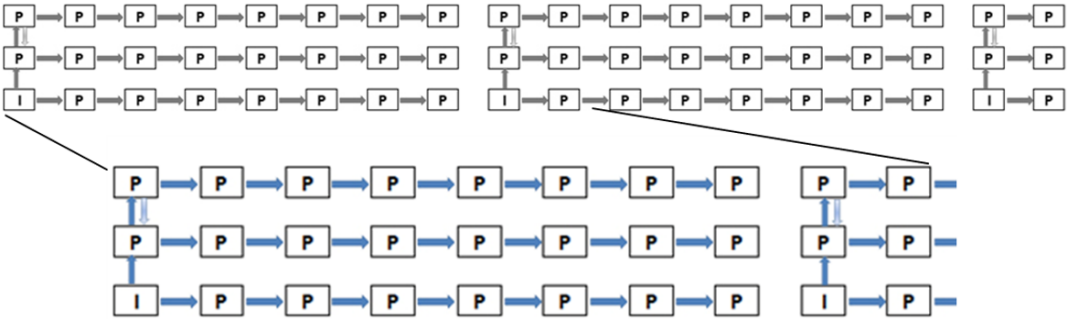


Figure 3.10: IPP Simplified Encoding Structure with GOP Size of 8 and with Three Views

IPP Full

In IPP Full encoding structure, both anchor and non-anchor (non-I-frame) frames are used to reference the other view. In this encoding structure, the stream size of referencing view becomes smaller due to the inter-view redundancy increases the search space. An example of the IPP Full encoding structure can be seen in Figure 3.11.

Hierarchical

The hierarchical encoding structure uses only the inter-view redundancy for only anchor frames on both views. As a consequence, all frames in the GOP are connected to each other

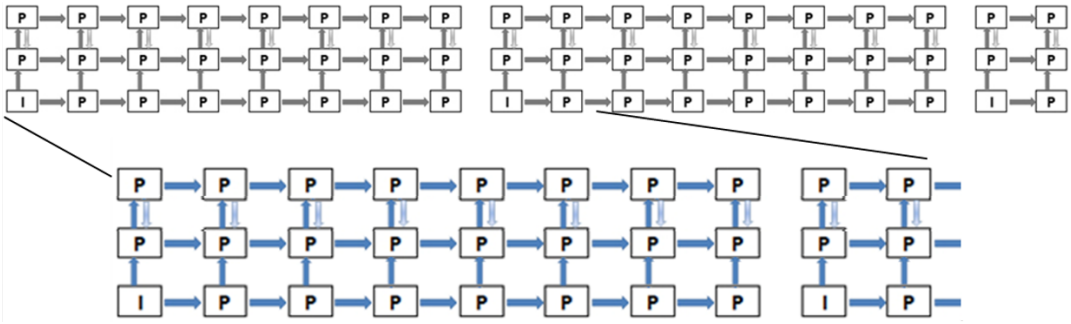


Figure 3.11: IPP Full Encoding Structure with GOP Size of 8 and with Three Views

due to the symmetric characteristics of the hierarchical encoding structure. The non-anchor frames are also referenced by the non-anchor frames of the other view in an indirect manner. An example of the hierarchical encoding structure in multiview encoding can be seen in Figure 3.12.

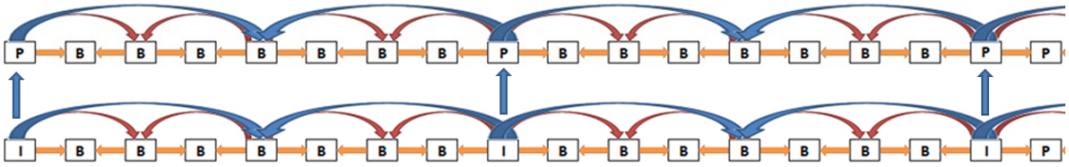


Figure 3.12: Hierarchical Encoding Structure with GOP Size of 8 and with Two Views

3.3 Experimental Setup

In order to find the most efficient encoding structure, there are two experiments conducted covering various 3D video compression methods and the performances of different encoding structures. The first experiment aims to compare three several stereo video encoding methods in terms of the encoding concepts. The second experiment aims to find an encoding method by looking the relation between the encoding methods and the encoding times of different encoding structures.

As it is mentioned, the main goal of the first experiment is to find an encoding method for the 3D video compression problem. For that purpose, a test set of video sequences has been chosen among the Mobile 3DTV WP1 outputs. The test set includes three different stereo video with depth. Properties of these videos are explained below.

For the experimental setup, the following encoding structures are chosen with respect to the video test set properties:

- H.264/MVC
- H.264/AVC Stereo SEI Message
- Video plus Depth Coding

The experiments are conducted on a desktop PC with Intel® Core™ i3-540 CPU (4 MB

Cache, 3.07 GHz), 2 GB DDRRAM, 32-bit Windows 7 OS and 180 GB HDD. The encoding and decoding is performed using JMVC 8.5. The configuration files and the necessary modifications are indicated in Appendix A. In this experiment, YUV color space is used as it is the most commonly used color space in the video encoding community. For the color subsampling of the videos, YUV4:2:0 is used, again one of the most common YUV subsampling methods in video processing [87].

HeidelbergAlleysR	
Resolution	432x240
Frame Rate	25 fps
Video Format	Raw, YUV4:2:0
3D Format	Left, Right
Content	Outdoor, city life, scene cuts, moderate object movement, moderate camera movement, high details, natural lighting, complex depth structures [74]. A frame is shown in Figure 3.13.
Creator	Dongleware. - http://www.dongleware.de/

Figure 3.13: A Snapshot from HeidelbergAlleysR Video Sequence

KnightsQuest	
Resolution	432x240
Frame Rate	30 fps
Video Format	Raw, YUV4:2:0
3D Format	Left, Right
Content	Computer animated film trailer, action, adventure, reasonable plot, multiple scene cuts and blends, various types of object motion, moderate to strong camera movement [74]. A frame is shown in Figure 3.14.
Creator	Red Star Studios. - http://www.redstarstudio.co.uk/

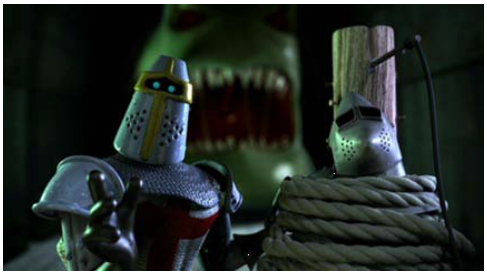


Figure 3.14: A Snapshot from KnightsQuest Video Sequence

RhineValleysMovingR	
Resolution	432x240
Frame Rate	25 fps
Video Format	Raw, YUV4:2:0
3D Format	Left, Right
Content	Nature, documentary, transportation, sports, outdoor, moderate and high camera and object movement, multiple scene cuts, various perspectives inclusive aerial views, high details, various depth complexities [74]. A frame is shown in Figure 3.15.
Creator	Cinovent. - http://www.cinovent.de/

Figure 3.15: A Snapshot from RhineValleysMovingR Video Sequence

3.4 Experimental Results

In this experiment, several 3D video encoding methods have been tested. In every 3D video compression methods, different encoding structures were tested. Each of these structures create a different bitstream. The main reason of the differences among these encoding structures is the different processing technique of each encoding structure.

In order to analyze the experimental results, first, we need to define the terminology used. A rate distortion (RD) curve is drawn by finding the bitrate and quality, or distortion, terms of each video stream. Bitrate is defined as the total bits over total seconds as it is shown in Equation 3.1.

$$Bitrate = \frac{Total\ bits\ of\ data\ stream}{Total\ frames} \times Frames\ per\ second \tag{3.1}$$

The distortion term, on the other hand, is defined as the logarithm of the ratio of peak signal to noise. The noise term is taken as Mean Squared Error (MSE).

$$MSE = \frac{1}{m \cdot n} \cdot \sum_{i=0}^{m-1} \times \sum_{j=0}^{n-1} [I(i,j) - K(i,j)]^2 \tag{3.2}$$

where the I is the original image, K is the distorted image, m and n are the sizes

$$PSNR = 10 \cdot log_{10}(\frac{MAX_I^2}{MSE}) = 20 \cdot log(\frac{MAX_I}{\sqrt{MSE}}) \tag{3.3}$$

The experimental results can be analyzed under the following three items:

- Multiview Video Coding
  - IPP Simplified
  - IPP Full
  - Hierarchical with GOP=8
  - Hierarchical with GOP=16
- Side-by-Side (SbS)
  - IPP
  - Hierarchical with GOP=8
  - Hierarchical with GOP=16
- Video plus Depth
  - IPP
  - Hierarchical with GOP=8
  - Hierarchical with GOP=16

In order to see the variation among different encoding structures, first, we should compare the results of all the encoding structures. For simplicity, result of only one channel is shown in each figure. This channel is mostly Y channel of the YUV color space. However, there are instances that Y channel does not show the realistic results and is subject to change due to some errors caused by encoding process or the video content. In those cases, V channel is shown instead of Y channel.

The figures numbered from 3.16 to 3.24 are the results of Multiview Video Coding (H.264/MVC) encoding method. In those figures, both left and right view and the overall quality computed by averaging PSNR values of left and right views are shown. In the figures numbered from 3.25 to 3.27, the results of H.264/AVC Stereo SEI message encoding method with side-by-side frame-packing type can be seen. Figures numbered from 3.28 to 3.30 are the results for Video plus Depth coding method.

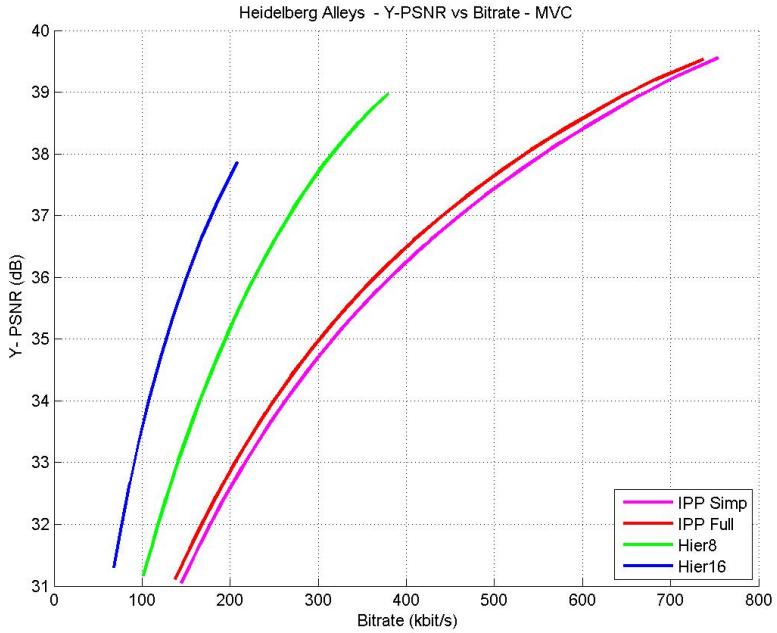


Figure 3.16: Y-PSNR vs Bitrate curves of Multiview Video Coding method for HeidelbergAlleysR sequence, Overall

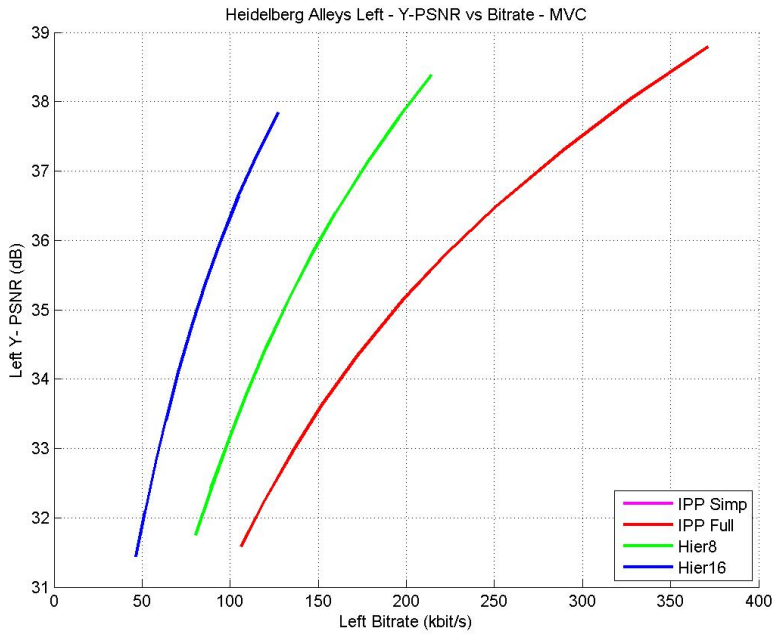


Figure 3.17: Y-PSNR vs Bitrate curves of Multiview Video Coding method for HeidelbergAlleysR sequence, Left view

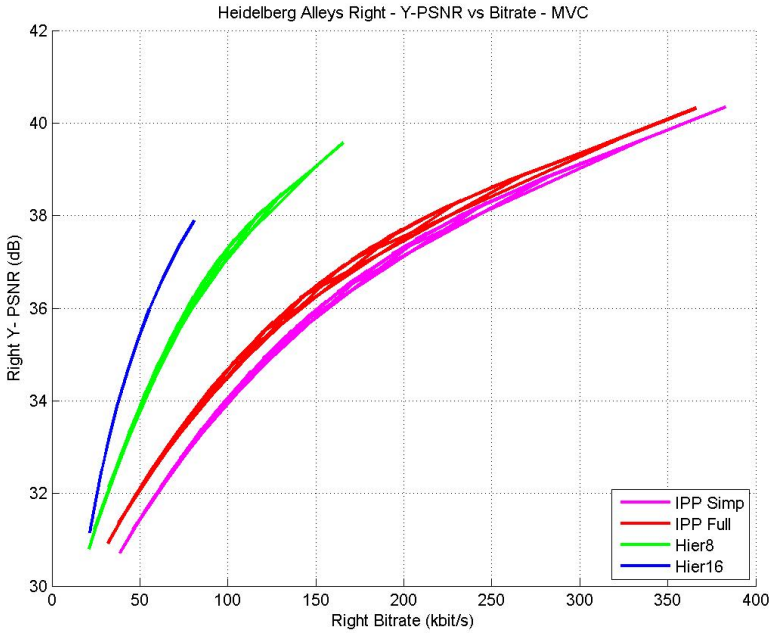


Figure 3.18: Y-PSNR vs Bitrate curves of Multiview Video Coding method for HeidelbergAlleysR sequence, Right view

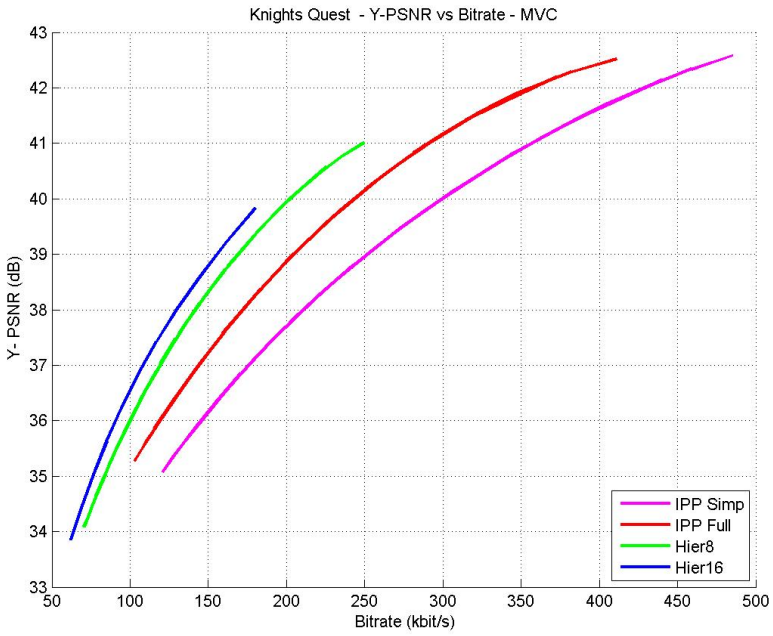


Figure 3.19: Y-PSNR vs Bitrate curves of Multiview Video Coding method for KnightsQuest sequence, Overall

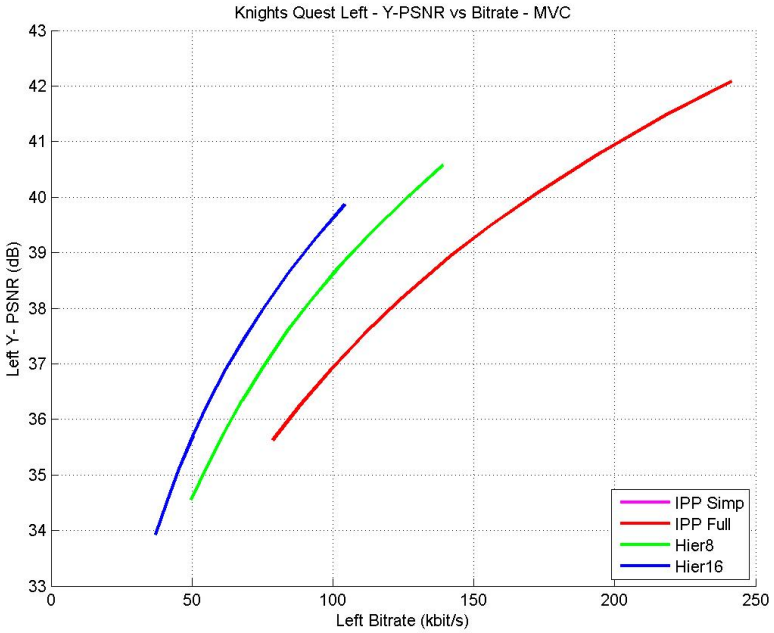


Figure 3.20: Y-PSNR vs Bitrate curves of Multiview Video Coding method for KnightsQuest sequence, Left view

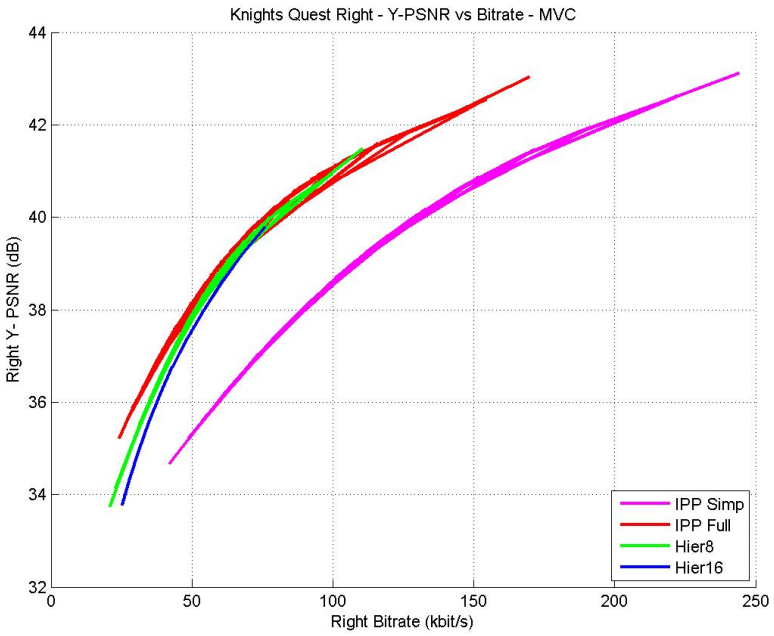


Figure 3.21: Y-PSNR vs Bitrate curves of Multiview Video Coding method for KnightsQuest sequence, Right view

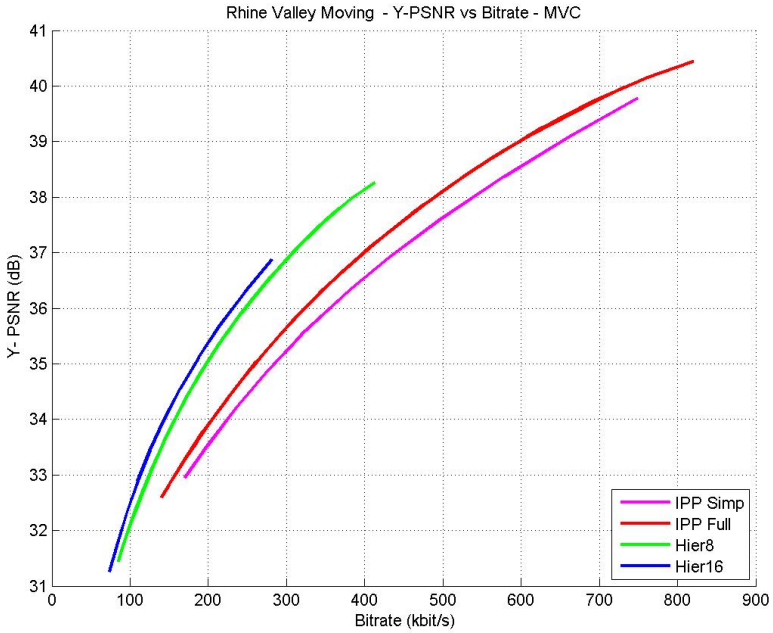


Figure 3.22: Y-PSNR vs Bitrate curves of Multiview Video Coding method for RhineValleysMovingR sequence, Overall

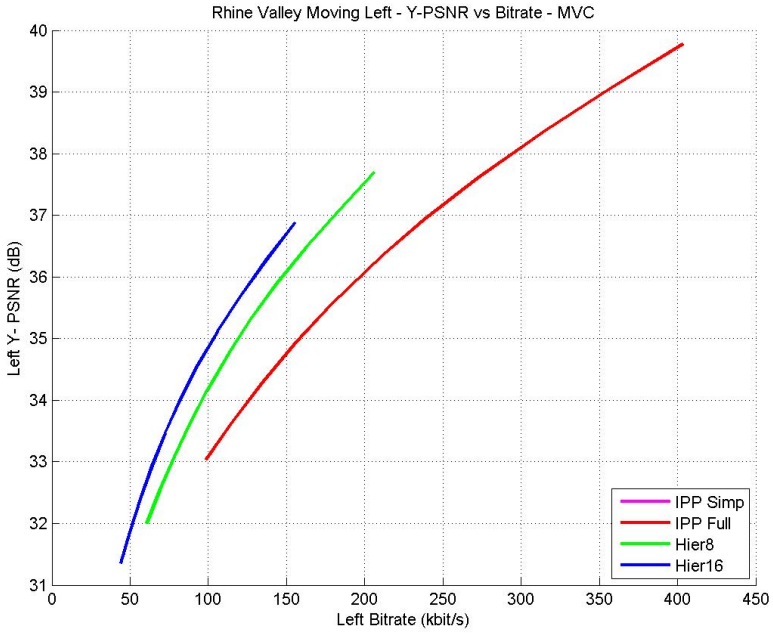


Figure 3.23: Y-PSNR vs Bitrate curves of Multiview Video Coding method for RhineValleysMovingR sequence, Left view

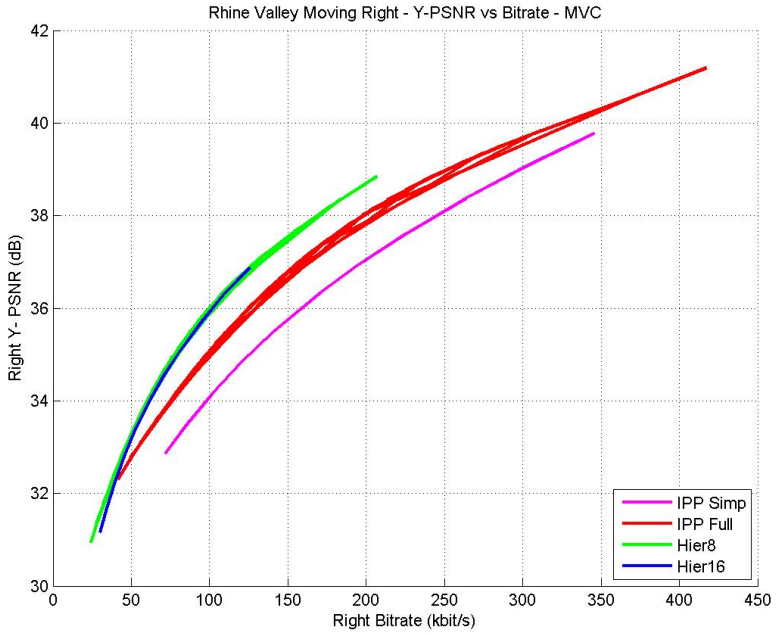


Figure 3.24: Y-PSNR vs Bitrate curves of Multiview Video Coding method for RhineValleysMovingR sequence, Right view

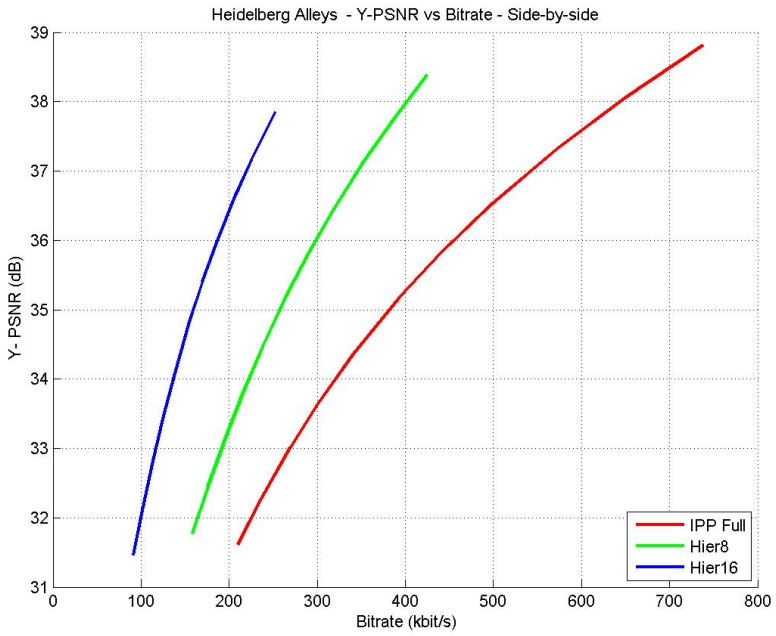


Figure 3.25: Y-PSNR vs Bitrate curves of H.264/AVC Stereo SEI message with side-by-side frame-packing method for HeidelbergAlleysR sequence

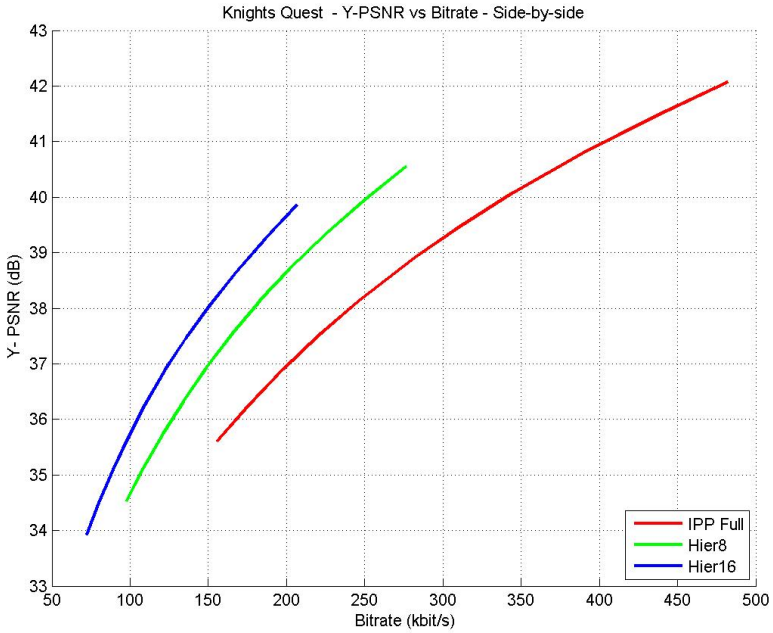


Figure 3.26: Y-PSNR vs Bitrate curves of H.264/AVC Stereo SEI message with side-by-side frame-packing method for KnightsQuest sequence

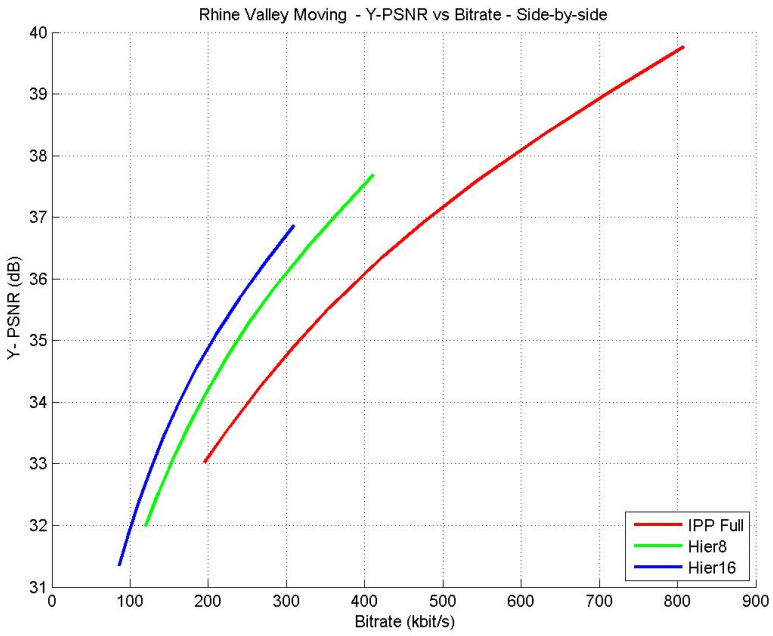


Figure 3.27: Y-PSNR vs Bitrate curves of H.264/AVC Stereo SEI message with side-by-side frame-packing method for RhineValleysMovingR sequence

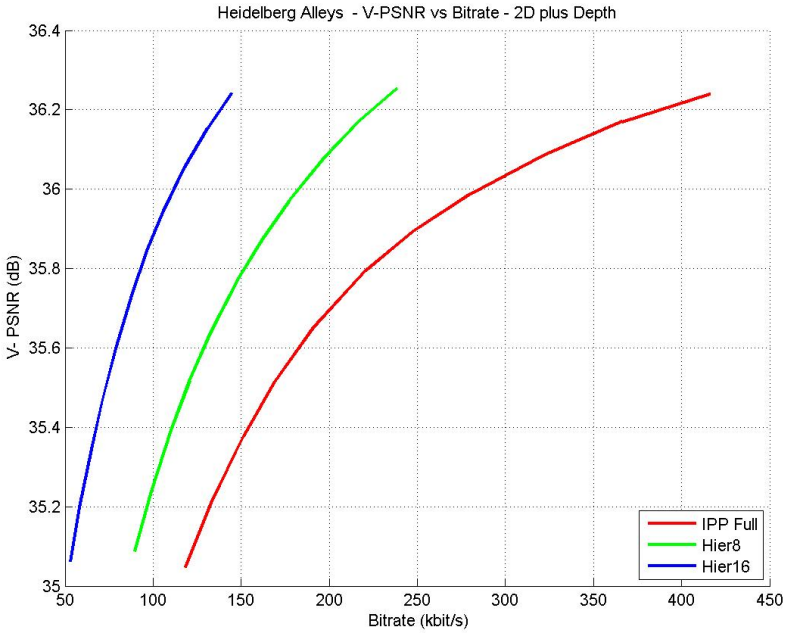


Figure 3.28: V-PSNR vs Bitrate curves of Video plus Depth method for HeidelbergAlleysR sequence

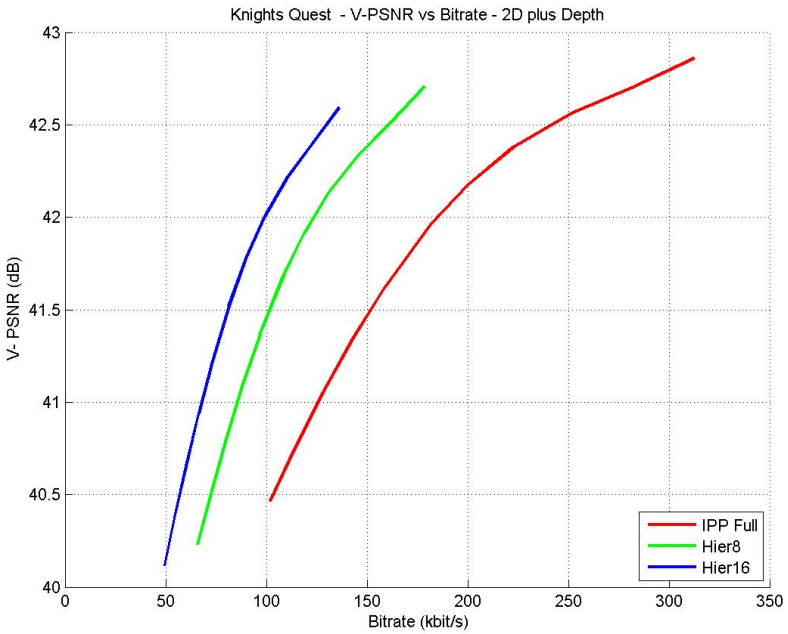


Figure 3.29: V-PSNR vs Bitrate curves of Video plus Depth method for KnightsQuest sequence

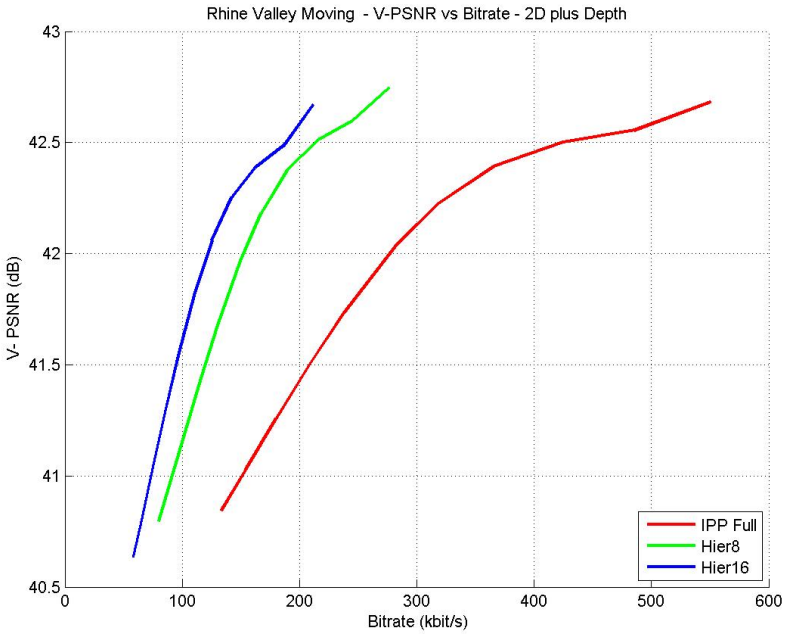


Figure 3.30: V-PSNR vs Bitrate curves of Video plus Depth method for RhineValleysMovingR sequence

3.4.1 Analysis of the Experimental Results

The overall RD curves for each video sequence with respect to the encoding structures are shown below in Figures 3.31 to 3.33. As it can be seen from the figures, video plus depth coding results have lower PSNR values than the other methods. Both this and being computationally complex on the receiver side, video plus depth coding method is not favorable. In order to find the most efficient coding method for mobile receivers, we need to consider also computational complexity. In order to measure that, the encoding times are presented in Tables 3.1-3.3.

By looking at the encoding times, we can say that the H.264/AVC Stereo SEI message method with side-by-side frame-packing is the simplest way of encoding. But that statement is not correct for the hierarchical structures. While side-by-side complexity is nearly equal as the MVC complexity in Hierarchical 8 GOP mode, the side-by-side method is more complex in Hierarchical 16 GOP mode than the MVC structure. The encoding complexity of the video plus depth is between MVC and SbS in every encoding structure. However, there is also a decoding and rendering complexity for video plus depth and it is not shown in these tables.

Considering these results, both side-by-side and MVC methods are more suitable for a system in which the receiver is mobile. Hence, these two methods were tested on an embedded transmitter system.

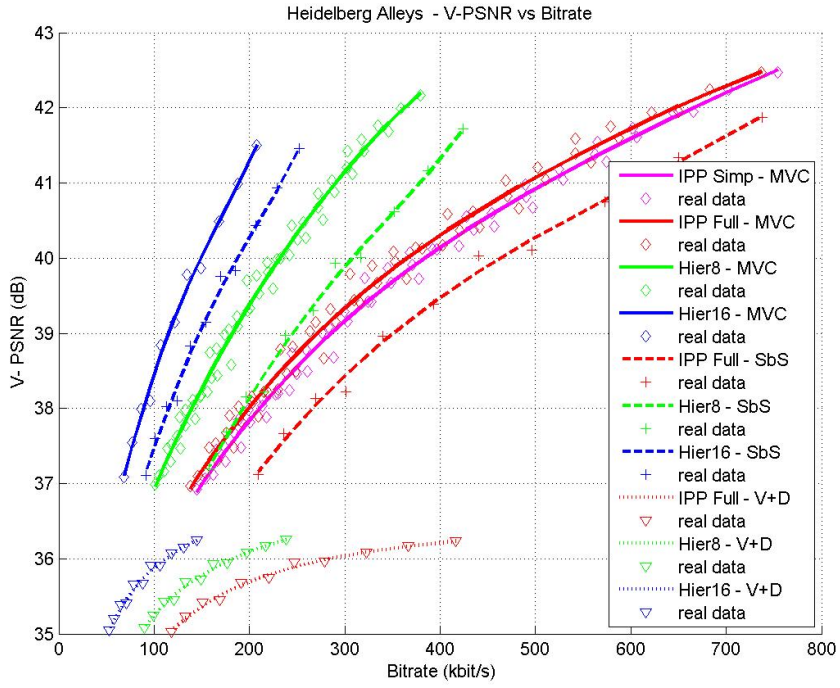


Figure 3.31: V-PSNR vs Bitrate curves of all encoding structures for HeidelbergAlleysR sequence, with real data

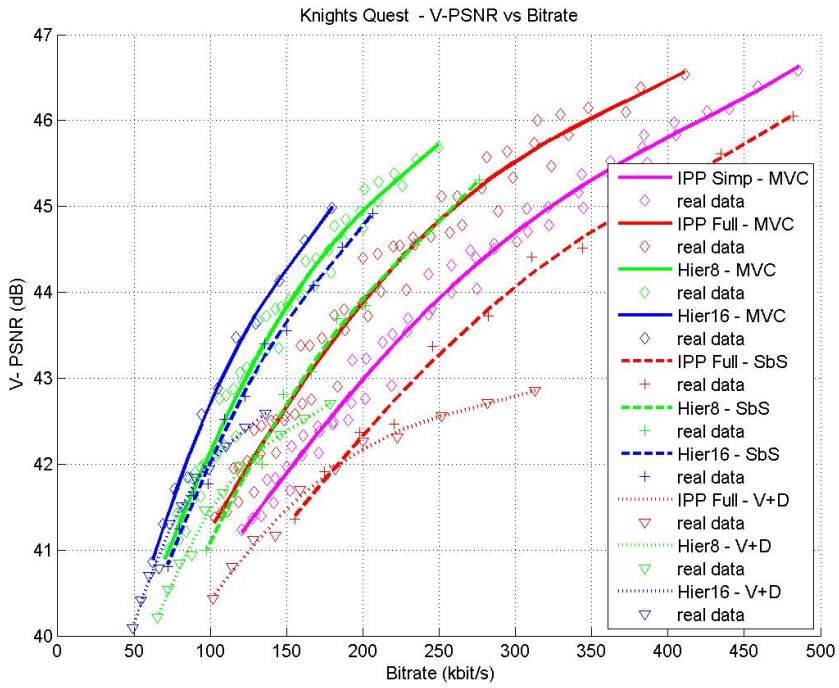


Figure 3.32: V-PSNR vs Bitrate curves of all encoding structures for KnightsQuest sequence, with real data

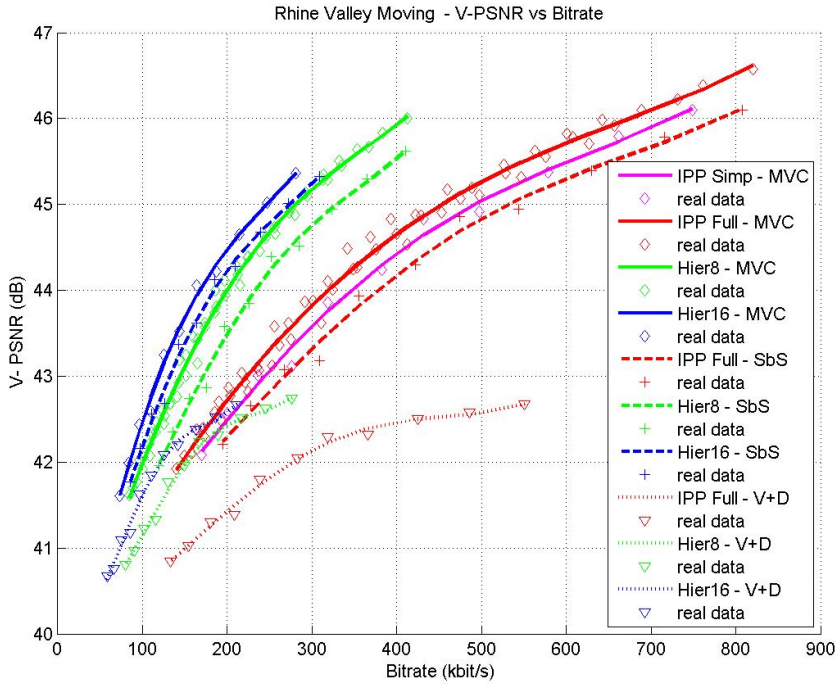


Figure 3.33: V-PSNR vs Bitrate curves of all encoding structures for RhineValleysMovingR sequence, with real data

Table 3.1: Encoding times of encoding structures for HeidelbergAlleysR sequence

Encoding times of HiedelbergAlleysR sequence (s)										
QP	MVC				Side-by-side			Video+Depth		
	IPP Sp	IPP Fl	Hier8	Hier16	IPP	Hier8	Hier16	IPP	Hier8	Hier16
26	369.29	584.60	2402.89	2451.70	342.43	2402.54	2873.43	346.53	2222.63	2672.89
27	366.23	576.74	2389.30	2442.01	340.46	2390.09	2856.28	338.02	2203.50	2657.08
28	363.26	572.22	2374.86	2426.99	339.53	2379.12	2837.91	333.34	2185.17	2630.65
29	360.85	566.26	2363.92	2422.85	334.43	2366.48	2837.94	330.85	2172.16	2621.35
30	359.00	560.46	2359.35	2405.78	331.99	2355.96	2812.13	326.59	2164.63	2598.12
31	354.30	553.17	2340.53	2390.66	332.43	2339.07	2793.94	323.36	2149.39	2579.49
32	352.33	548.14	2329.86	2383.25	327.63	2326.80	2782.91	319.41	2138.17	2572.16
33	349.49	541.38	2324.68	2374.30	327.52	2323.77	2775.14	317.28	2131.42	2562.12
34	347.09	535.68	2310.79	2360.78	323.28	2309.84	2759.31	313.94	2120.55	2546.08
35	345.34	528.58	2303.31	2350.54	324.75	2299.61	2743.36	310.40	2112.59	2536.44
36	342.06	524.67	2285.06	2335.28	319.54	2283.86	2734.56	308.77	2099.26	2523.09

Table 3.2: Encoding times of encoding structures for KnightQuest sequence

Encoding times of KnightQuest sequence (s)										
QP	MVC				Side-by-side			Video+Depth		
	IPP Sp	IPP Fl	Hier8	Hier16	IPP	Hier8	Hier16	IPP	Hier8	Hier16
26	408.79	664.99	2636.90	2748.96	390.97	2664.27	3168.85	398.39	2471.22	2978.43
27	405.98	657.56	2625.42	2721.57	385.06	2642.61	3137.78	384.23	2448.95	2946.37
28	403.33	647.64	2599.01	2699.42	381.89	2611.90	3103.02	374.72	2421.68	2914.13
29	400.21	639.07	2573.22	2666.25	379.22	2585.93	3072.86	371.66	2395.69	2881.48
30	396.14	630.24	2551.15	2636.12	376.76	2562.69	3040.71	367.97	2374.23	2851.97
31	394.25	623.15	2522.02	2607.41	371.74	2534.54	3005.13	362.70	2346.11	2818.88
32	390.46	614.59	2501.51	2581.83	369.90	2514.08	2976.42	357.56	2325.16	2792.34
33	386.52	604.53	2475.83	2551.93	364.12	2493.03	2945.09	352.88	2301.73	2765.91
34	383.46	595.24	2451.58	2521.08	361.49	2472.56	2913.71	349.71	2281.36	2736.58
35	380.06	599.48	2427.09	2488.35	359.40	2448.43	2880.43	347.85	2259.29	2709.00
36	377.78	587.17	2397.12	2460.89	354.01	2416.52	2846.51	342.06	2235.46	2683.03

Table 3.3: Encoding times of encoding structures for RhineValleysMovingR sequence

Encoding times of RhineValleysMovingR sequence (s)										
QP	MVC				Side-by-side			Video+Depth		
	IPP Sp	IPP Fl	Hier8	Hier16	IPP	Hier8	Hier16	IPP	Hier8	Hier16
26	421.81	714.98	2806.01	2924.11	393.55	2842.46	3375.58	418.61	2574.77	3084.25
27	418.98	703.63	2771.89	2883.48	390.00	2812.06	3323.08	409.49	2531.95	3033.41
28	412.67	692.84	2731.34	2839.20	387.99	2772.19	3265.93	399.88	2488.87	2983.61
29	408.59	681.36	2691.68	2797.66	381.79	2732.51	3226.03	390.69	2451.79	2941.72
30	403.78	671.79	2654.85	2752.35	378.07	2691.16	3168.81	384.16	2416.09	2895.98
31	401.15	661.69	2610.76	2703.15	375.67	2649.94	3112.89	377.02	2379.69	2850.48
32	396.89	649.03	2571.38	2660.37	369.98	2606.67	3066.12	367.18	2346.25	2817.99
33	392.96	634.86	2533.54	2618.80	369.22	2572.13	3017.20	361.80	2313.10	2778.59
34	389.05	624.70	2497.42	2572.05	362.90	2531.13	2966.43	354.27	2287.58	2741.97
35	385.86	611.17	2458.27	2525.94	359.71	2492.73	2917.32	351.05	2258.00	2706.38
36	382.64	600.33	2424.33	2484.67	357.55	2449.41	2870.24	346.70	2228.78	2670.56

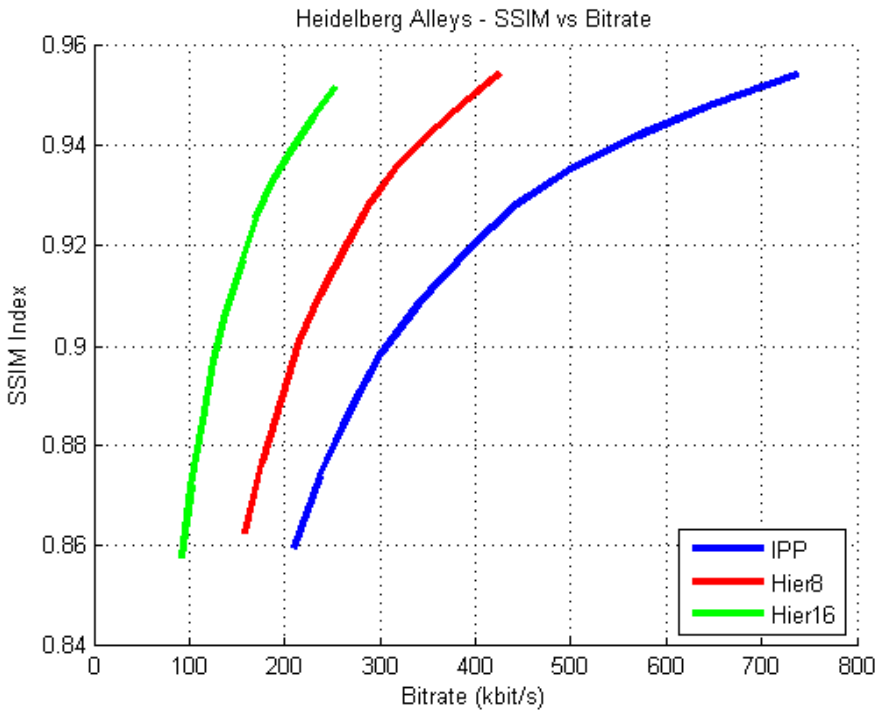


Figure 3.34: SSIM vs Bitrate curves of Side-by-Side encoding structure for HeidelbergAlleysR sequence

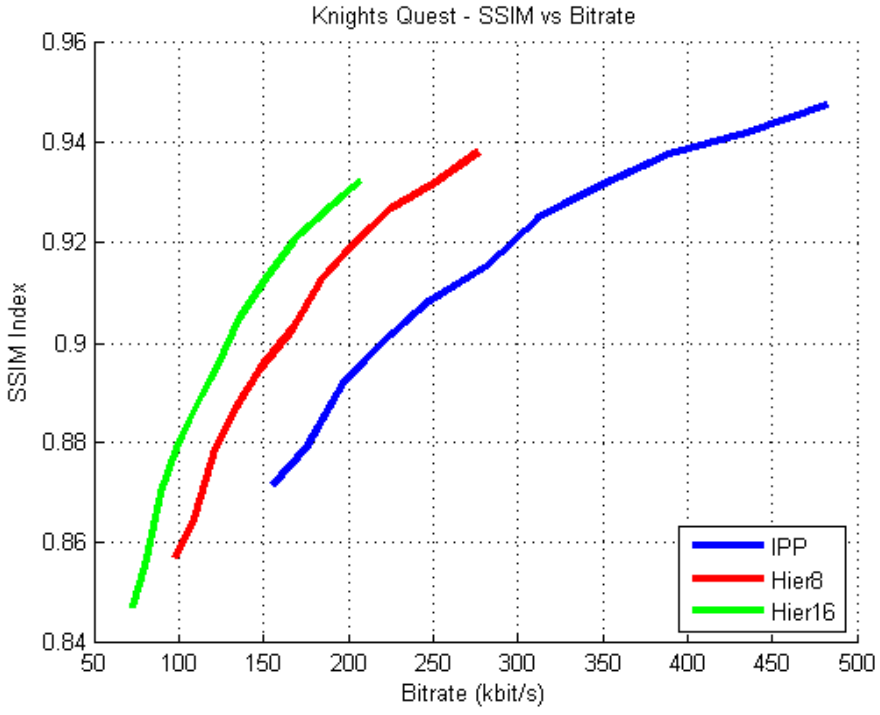


Figure 3.35: SSIM vs Bitrate curves of Side-by-Side encoding structure for KnightsQuest sequencea

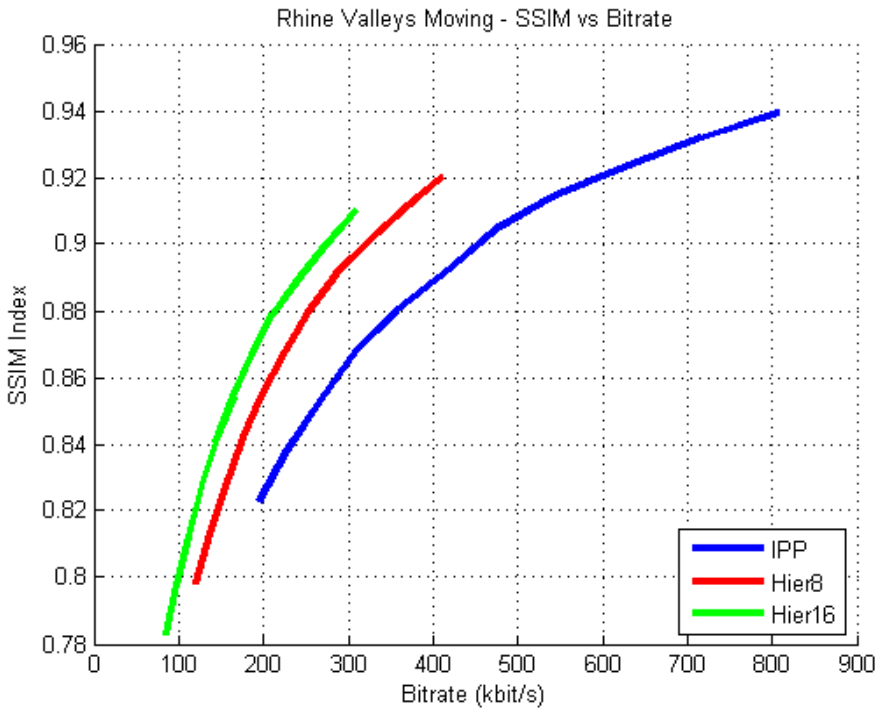


Figure 3.36: SSIM vs Bitrate curves of Side-by-Side encoding structure for RhineValleysMovingR sequence

Despite the fact that the measurement of quality is made in terms of PSNR, also different quality measures can be used to find rate distortion curves. In Figures 3.34-3.36, the SSIM (Structural Similarity Index) vs Bitrate for only Side-by-Side encoding method is shown in order for a comparison between the PSNR and SSIM quality measurement metrics. Both the PSNR and SSIM agree on the results of different encoding methods even though SSIM does not agree with PSNR in many different occasions.

Considering the figures, some implications about the performances of the encoding methods can be made. First of all, Video plus Depth encoding method gives the worst PSNR value. Even though the solid change in the PSNR value is caused by the rendering issues, the given quality values renders the V+D encoding unsuitable. When compared, the results of MVC and Side-by-Side/AVC encoding methods are very similar, and they change with the encoding structure mostly. The encoding times of both MVC and SbS/AVC give very similar results also. Even though the MVC and SbS/AVC methods give similar results, MVC needs a special encoder and decoder in order to handle the compression and decompression tasks whereas the SbS/AVC encoding complies with the previous encoders/decoders and does not need a specialized software.

By considering both the encoding performance as the rate of compression and the encoding times, the MVC and SbS/AVC encoding methods give good results. However, regarding the MVC needs specialized software, Side-by-Side gains an advantage on the applicability to the systems with plain H.264 decoder.

3.5 Experimental Setup on BeagleBoard-xM

This experiment is prepared to analyze the performance of the embedded transmitter system under different encoding structures. For that, the side-by-side frame-packing representation method is chosen due to its easiness and popularity among the commercial multimedia device producers. MVC is not included in this experiment since currently there is not any MVC encoder/decoders in Linux, which is the operating system of Beagleboard.

In this experiment, BeagleBoard-xM is used as the transmitter embedded program. The BeagleBoard-xM has an ARM and DSP core DM 3730 System on Chip processor. Full list of specifications can be found in Appendix B. The proposed system is based on 1-GHz ARM® Cortex™-A8 processor and 512 MB DDRRAM. In the experiment, an open source video encoder based on H.264/AVC encoding, *"x264"*, is used. *x264* is created by Videolan organization and it is integrated within the open source streamer and media player *VLC Media Player*, which is also created by Videolan organization. The encoding properties and variables can be seen in the sample command below.

```
$ x264 --profile main --keyint 8 --bframes 0 --qp 26 --frame 752 --psnr --me
  dia --subme 1 --no-chroma-me --merange 32 --ref 4 --deblock 0:0 --weightp 0
  --no-weightb --no-cabac --no-progress --output [outputFile] [inputFile]
  [resolution]
```

3.5.1 Experimental Results on BeagleBoard-xM

The results of the experiment on BeagleBoard-xM is given in Figures 3.37 - 3.39. Timing results of this experiment is given in Table 3.4. For the case of bitrates, performance of x264 is lower. However, there is not much difference in the quality side. When the timings are compared, the Hierarchical with 8 GOP performs nearly as good as IPP. Hence, the Hierarchical encoding structure could be chosen instead of IPP.

Table 3.4: Encoding times of SbS encoding structures for all sequences. TET indicates "Total Encoding Time", and FpS indicates number of frames encoded per second

Encoding times for Side-by-side encoding method on x264 on BeagleBoard-xM (s)												
QP	HeidelbergAllesyR				KnightsQuest				RhineValleysR			
	IPP		Hier8		IPP		Hier8		IPP		Hier8	
	TET	FpS	TET	FpS	TET	FpS	TET	FpS	TET	FpS	TET	FpS
26	62.73	12.04	60.02	12.50	70.19	10.75	84.21	8.92	119.11	6.32	117.39	6.41
27	58.30	12.82	58.56	12.82	73.42	10.20	76.41	9.80	121.23	6.21	116.43	6.45
28	56.53	13.33	56.71	13.33	67.29	11.23	85.83	8.77	113.89	6.62	116.78	6.45
29	56.74	13.33	51.62	14.49	70.72	10.63	86.75	8.69	115.06	6.53	116.00	6.49
30	53.63	14.08	59.30	12.65	71.81	10.52	86.80	8.69	115.23	6.53	114.03	6.57
31	51.98	14.49	53.75	14.08	76.18	9.90	87.65	8.54	114.04	6.57	135.05	5.55
32	52.60	14.28	55.15	13.69	69.19	10.86	87.57	8.62	115.37	6.53	131.58	5.71
33	49.99	15.15	51.48	14.70	79.58	9.43	85.45	8.77	109.82	6.84	126.75	5.91
34	51.11	14.70	53.29	14.08	69.22	10.86	88.38	8.47	109.93	6.84	130.71	5.74
35	48.00	15.62	53.00	14.28	79.36	9.43	90.41	8.33	108.65	6.94	118.43	6.36
36	47.33	15.87	52.63	14.28	71.27	10.52	88.45	8.47	97.82	7.69	103.72	7.24

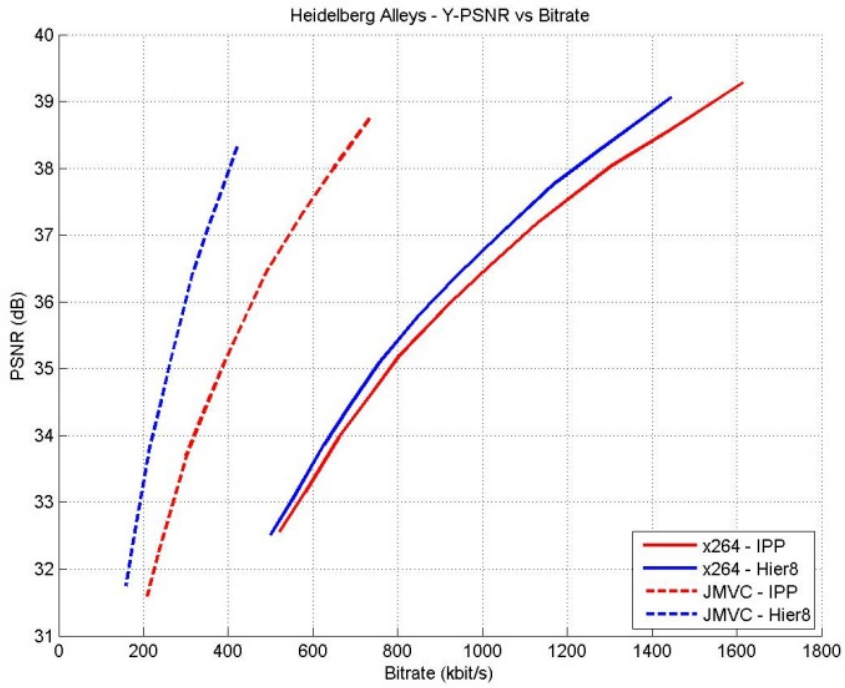


Figure 3.37: Y-PSNR vs Bitrate curves of Side-by-Side encoding structures for HeidelbergAlleysR sequence, with JMVC data for comparison

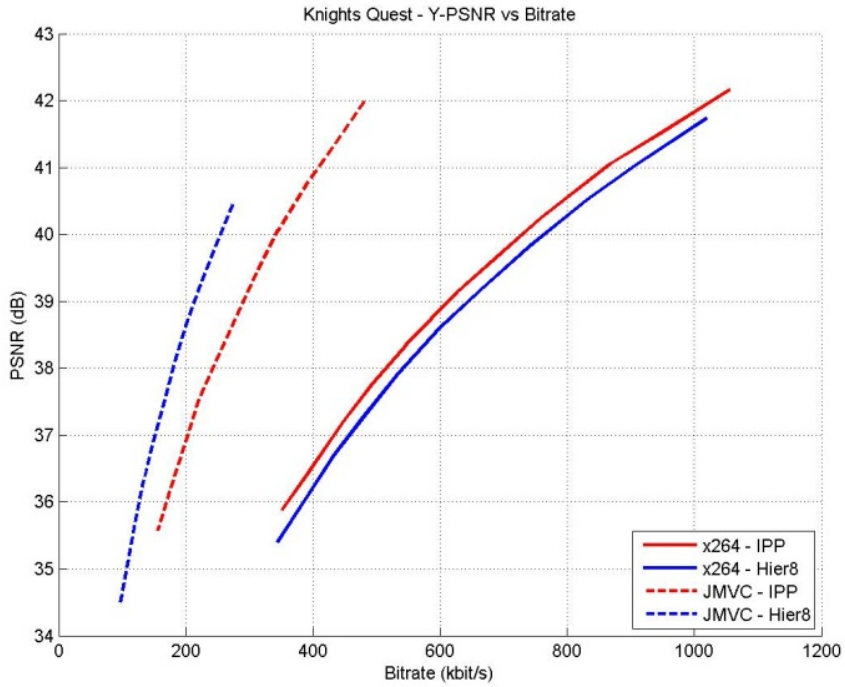


Figure 3.38: Y-PSNR vs Bitrate curves of Side-by-Side encoding structures for KnightsQuest sequence, with JMVC data for comparison

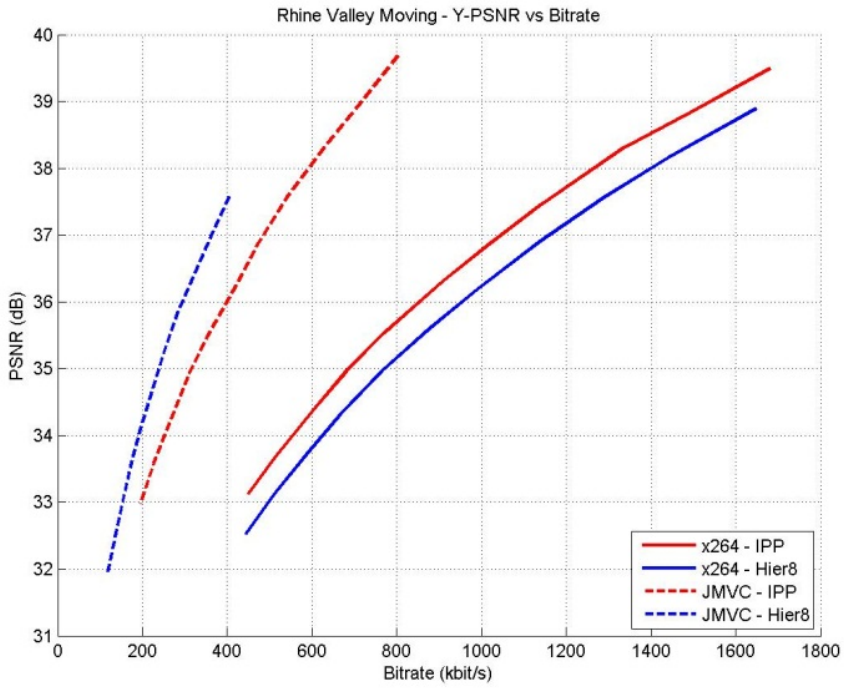


Figure 3.39: Y-PSNR vs Bitrate curves of Side-by-Side encoding structures for RhineValleysMovingR sequence, with JMVC data for comparison

By looking the experimental results of the BeagleBoard-xM, the hierarchical encoding structure is chosen as the encoding structure of the H.264/AVC Simulcast with Stereo SEI Message with Side-by-Side frame representation due to both the bitrate advantage and the small encoding time difference compared to the IPP encoding structure.

CHAPTER 4

VIDEO STREAMING

Video transmission is an important problem in multimedia systems. In an end-to-end system, the weakest link decides the video quality, and the transmission step is the most vulnerable step of an end-to-end system due to the losses or physical damages. There are different video transmission methods used throughout this century. The most basic video transmission method is transmitting video in solid media such as electromagnetic films and optical devices such as VCD or DVD. The other basic transmission method is broadcasting the video. In this method, primarily analog video data is multiplexed onto electromagnetic waves by using different AM or FM techniques. Digital video broadcasting methods are comparably new, and they use different digital multiplexing strategies to deliver digital media. Beside these methods, the internet video streaming systems are used on the packet switched network. All of these video transmission methods can be separated as streaming, such as broadcasts, and non-streaming. In this thesis, non-streaming transmission methods are out of scope.

In this chapter, different video streaming methods will be discussed and compared in order to find a suitable video streaming method for the proposed system. These video streaming methods include digital television broadcast methods and different video streaming methods based on different network protocols.

4.1 Methods for Video Streaming

Video streaming can be in different types and on different physical or logical layers. When it is looked to the video streaming, the most known example, TV, used analog transmission for decades. The analog transmission is based on signal modulation, and this transmission uses different landlines and air as its media. Different standards are generated such as PAL, NTSC, and SECAM throughout the world. AM or FM modulation was used for TV broadcasting, FM being the most common. After 1990s [88, 18], digital broadcasting techniques began to emerge. The digital broadcasting systems use digital transmission techniques. The most commonly used technique is the digital electromagnetic wave modulation method which has different modulation types such as Quadrature Phase-Shift Keying (QPSK), Quadrature Amplitude Modulation (QAM) and Orthogonal Frequency Division Multiplexing (OFDM).

With the increase in both the digital video creation and the number of digital video transmission methods, the Internet became also a major component in the digital video streaming topic. The Internet can be accessed by many people using different methods of connections

such as dial-up connections, Digital Subscriber Line (DSL) applications, and Cable transmission. The packet-switching structure of the internet enabled a robust video transmission system. Also, different techniques such as Video on Demand (VoD) applications made it possible to access videos anytime regardless of broadcast times.

Digital video streaming can be divided into two main topics; the digital broadcast and the Internet-based streaming methods. The digital broadcast methods were developed by associations formed by government-based institutions and private companies at the beginning of the 1990s in different parts of the world. These methods include Advanced Television Systems Committee (ATSC) in USA, Digital Video Broadcasting (DVB) in Europe, Integrated Services Digital Broadcasting (ISDB) in Japan, Digital Terrestrial Multimedia Broadcast (DTMB) in China and Digital Multimedia Broadcasting (DMB) in South Korea.

Most digital television systems use the same or very near frequency interval; however, they use different digital signal modulation methods such as OFDM, QAM, and QPSK. There are Internet Protocol (IP) compliant digital television standards such as DVB-IPTV. These digital television standards also include mobile transmission standards named as ATSC/M-H, DVB-H and DMB. These transmission methods are used in several applications, and these applications are discussed in Chapter 5.

4.2 Methods for Video Streaming over the Internet

Digitally stored video can be transmitted over the Internet in several ways. The compressed or raw video should be packetized into small network units and these small units must be reconstructed at the end of the transmission line. In order to reconstruct the packetized and transmitted video network units, a special software or programming needed. For these purposes there are different methods. The main difference between these methods is the way of the client to communicate the server, also known as communication protocol. There are different network protocols in both the application level and transport level that differs the video streaming methods over the Internet.

4.2.1 Network Protocols

In order to understand the streaming protocols, one must know the network interface structure. The network is designed to work in different layers, and this layered structure helps both applications to work in a network friendly way and different applications to communicate each other easily. There are different simplifications on the layers of the network. However, the most accepted one is the TCP/IP Network Model or 5-layer reference model [83]. This model consists of 5 different network layers, namely application layer, transport layer, internet layer, link layer (or data-link layer) and physical layer. The most known examples for these layers are Hypertext Transport Protocol (HTTP) for application layer, Transmission Control Protocol (TCP) for transport layer, Internet Protocol (IP) for internet layer, Address Resolution Protocol (ARP) for link layer and Ethernet with twisted pair cable for physical layer.

The transmission of data in layered structure works as following as it is shown in Figure 4.1. The big data or message of the application is prepared by the application protocol and passed

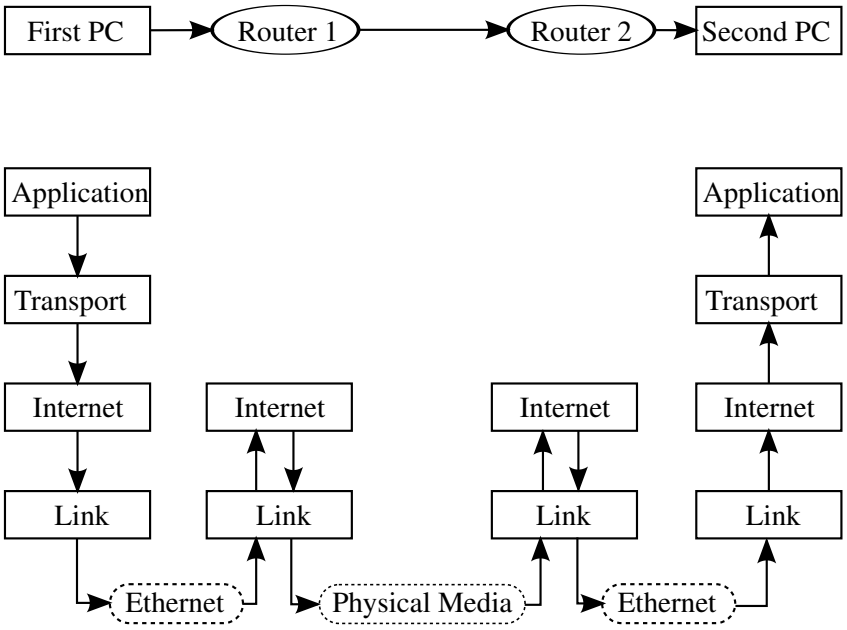


Figure 4.1: Network Topology and the Data Flow Structure between Two Devices

to the transport protocol. Transport protocol finds the required network devices and handles the network as adding its header and managing a seamless transport channel between two devices and passes the packet to the internet layer. Internet layer handles the routing of the packet and connectivity to other devices such as routers as adding its own header. The link layer manages packet transmission between devices connected to the same local network and handles the crosstalk among these devices.

Different network protocols offer different services and there are different video streaming methods. These methods can be counted as UDP, TCP, RTP, RTSP, HTTP and DASH. All these video streaming methods are based on the Internet Protocol (IP). There are also different protocols which are used in different systems such as Internet Group Management Protocol (IGMP) on the Internet level used for IPTV applications [99], and Scalable Trasmission Control Protocol (STCP) and Even-to-Sink Reliable Transport (ESRT) used for wireless sensor networks [52]. However, these systems are not very useful for the mobile transmission problems.

There are different aspects of video streaming such as the target audience size, the channel being static or dynamic, video stream being real-time or pre-recorded, and the connection type being packet-switched or circuit-switched network. The target audience size decides whether the system is needed to be whether unicast, multicast, or broadcast. Being static or dynamic channel changes the type of connection from constant bitrate to variable bitrate. Real-time of prerecorded video property alters the streaming priorities and changes the streaming type as well as the connection type. All these aspects add different special requirements to the video streaming problem. These requirements are discussed in Section 4.3.

4.2.2 User Datagram Protocol (UDP)

User Datagram Protocol is a transport layer network protocol. It is stateless, unidirectional and easy to implement. However, UDP does not guarantee the delivery of the packet to the receiver side as it is unidirectional. These properties of UDP makes is favorable for simple network assignments which does not require the exact delivery and also needed to be fast. These characteristics of the UDP makes it an advantageous broadcast protocol for different applications and especially real-time multimedia delivery.

The UDP Unicast was the most popular real-time multimedia delivery protocol for a long time [6, 34]. UDP is even used as transport layer network protocol for different Peer to Peer (P2P) streaming systems[69]. Also, it created a base for other multimedia streaming protocols. However, UDP does not have any feedback mechanisms. Hence, advantageous mechanisms such as flow control and congestion control are not included in UDP streaming.

4.2.3 Transport Control Protocol (TCP)

Transport Control Protocol is also a transport layer network protocol. Different from UDP, TCP supports several additional utilities which makes TCP a safe protocol; however, vulnerable to delay due to its unbounded retransmission time. These utilities are:

- Guarantee of the packet delivery
- Flow control and congestion control
- Feedback channel coming from the receiver

Due to the advantages of the TCP against UDP, TCP is used in Video on Demand (VoD) applications dominantly where the general quality is more important than the real-time transmission.

In early works, TCP was thought not to be the best network protocol to implement in real-time multimedia delivery problems due to the change in temporal conditions caused by flow and congestion control and the unbound retransmission time [34, 46, 82]. On the other hand, Kuschnig *et al.* stated that "TCP provides satisfactory performance" in 2010. The developments in the high bandwidth internet connections, introduced delay of TCP has less importance than the advantages of the TCP [39, 8, 65].

4.2.4 Real-Time Transport Protocol (RTP)

Real-Time Transport Protocol is an application layer network protocol which is created for the transmission of especially audio and video files. RTP is created by Audio Video Transport Working Group of Internet Engineering Task Force (IETF). The specifications of the protocol is given in several Request for Comments (RFC) [67, 26]. RTP is designed as a protocol to cover the multimedia transmission process in an end-to-end manner.

RTP is a commonly used streaming protocol. Mostly, UDP is utilized under RTP [1, 44, 98]. RTP/UDP/IP, or RTP/UDP, system is able to create a multicast streaming due to the

unidirectional structure of the UDP [47]. In this configuration, the advantages of TCP such as flow control, congestion control and feedback channel cannot be used. In order to decrease this shortcoming, there are some implementations of TCP-friendly UDP methods [1]. TCP can also be used under RTP; however, that is not a commonly applied method.

RTP Control Protocol (RTCP) is used to control the status of the transmisson and carry feedback information in order to find the Quality of Service (QoS) parameters and synchronization of the multiple streams. Mostly RTCP and RTP are used together in order to control the streaming process [1, 6, 90].

4.2.5 Real-Time Streaming Protocol (RTSP)

Real-Time Streaming Protocol is an application layer network protocol which is created for real-time multimedia streaming and session handling between transmitter and receiver. RTSP is a protocol for managing and controlling the streaming process. RTSP does not involve in transmission itself. Most of the RTSP implementations utilize RTP and RTCP for streaming purposes. RTSP starts a session with the client and maintains this session, or state, to the end of the streaming process. Besides, RTSP also offers different functions such as PLAY, PAUSE, SEEK, and RECORD [6]. RTSP is used in different systems such as IPTV applications and real-time video transmission for remote controlling of a wheeled system [34, 44, 77].

4.2.6 Hypertext Transfer Protocol (HTTP)

Hypertext Transfer Protocol is an application layer network protocol that is used to transmit different kinds of information. HTTP is a commonly used network protocol and it is created by IETF [24]. HTTP is mostly used in request-response management for different applications such as browsing the web and connecting the servers to request to download the web pages. Even though UDP can also be used, HTTP often uses TCP transport protocol.

In previous studies, HTTP is not mentioned even in a detailed comparisons among the video streaming methods [6]. In the beginning, HTTP was used for progressive download of the videos stored on the web. These videos can be decoded and watched while downloading the rest of the content. However, HTTP was not capable of live video streaming [34]. Today, HTTP is the current trend in most of the streaming, especially web streaming, applications [8].

The video streaming on HTTP is similar to the structure of RTSP. HTTP only handles the request and response messages and the transmission is carried out by TCP. The main difference between HTTP and RTSP is that server keeps state information of the client in RTSP whereas server does not keep states in HTTP, instead, the client keeps its own state [78].

For the development of the Adaptive HTTP Streaming, different companies came out with their standards. The most important and known adaptive HTTP streaming protocols are Apple's HTTP Live Streaming (HLS), Microsoft's Live Smooth Streaming (LSS), and Adobe's HTTP Dynamic Streaming (HDS) [8, 65]. These protocols work in a similar way. Each of them stores encoded videos in their own formats and most importantly in different fragments. These fragments are requested in parallel to the client's computations in order to

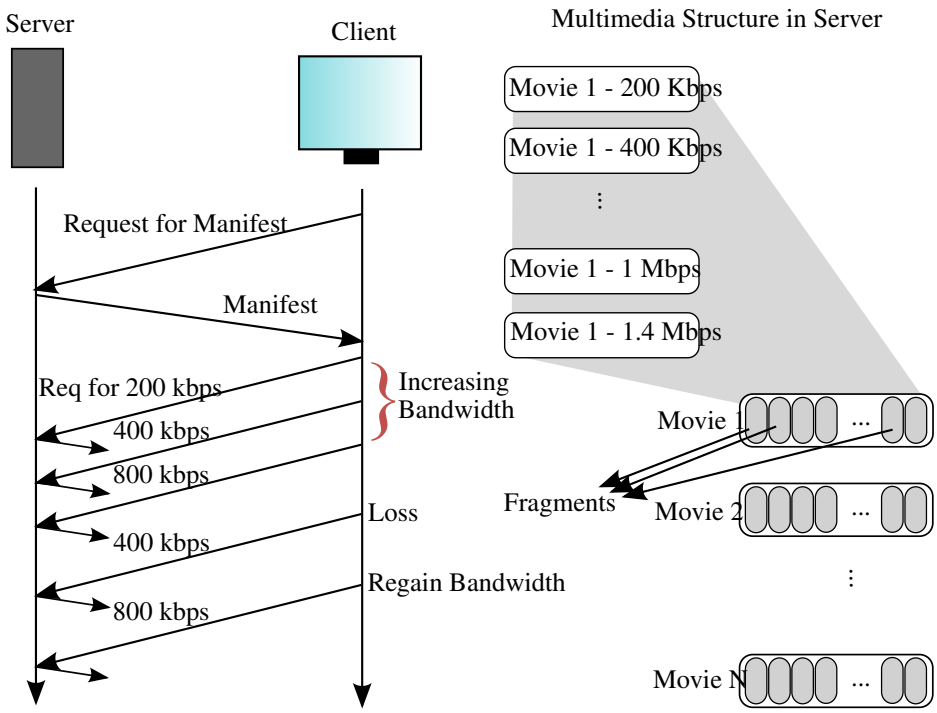


Figure 4.2: An Example of Adaptive HTTP Streaming Client Process [8]

supply best view to the user. If there is a network congestion and not all of the packets are arriving to the decoder buffer, the client's adaptive HTTP streaming protocol requests for a low bitrate fragment as it is shown in Figure 4.2.

MPEG community released a new HTTP adaptive streaming standard in April 2012 named as MPEG Dynamic Adaptive Streaming over HTTP (DASH). The aim of development of DASH is to create an international standard by using the acquired know-how of the adaptive HTTP streaming [78]. Similar to HLS, the change in fragments or *media segment*s are done by the MPEG DASH client and the video bitrate changes seamlessly due to the structure of the *representation*s.

4.3 Analytical Comparison

Video streaming topic has different requirements for each different video streaming application. In order to compare streaming protocols, these requirements have to be considered. These factors can be described as follows:

- **Audience size:** Related to stream type, unicast, multicast, or broadcast
- **Channel properties:** Static bitrate channels can be handled by constant bitrate solution; while a dynamic bitrate channel needs adaptive bitrate
- **Stream type:** Real-time videos need to be transmitted by a network protocol different than the pre-recorded videos

Table 4.1: Comparison of Different Packet based Video Streaming Protocols on the Internet

	Multicast	QoS support	Adaptivity	Real-Time	VoD	Descriptions for suitability
UDP	✓	-	-	✓	-	Doesn't have a feedback channel
TCP	-	-	-	✓	✓	Works with RT videos with small delay
RTP	✓	✓	-	✓	-	
RTSP	✓	✓	✓	✓	-	
HTTP	-	✓	✓	✓	✓	Works with RT videos with small delay, can traverse firewalls

- **QoS support:** Whether the streaming protocol provides QoS support by sending parameters related to network and the receiver or not

Apart from these factors, a network protocol should be compared within the environmental condition such as the bandwidth requirements or suitability to the targeted application.

Considering these factors, a comparison should be made between the Internet-based streaming methods. All applications can work on real-time with a note that TCP and HTTP works with more delay than the UDP/RTP counterparts, this is important. TCP and HTTP can not multicast; however, this is not a big flaw in our desired system. UDP and TCP do not have QoS support and do not have adaptivity possibilities like RTP. Only TCP and HTTP are reliable when it comes to VoD support. However, this is not important in our desired system.

In proposed system, with respect to network behavior the most important aspects are the trans-layer adaption abilities of stream packets in the network structure and the adaptivity to the network congestion. Both of these factors are related to the mobility of the devices. Receiver mobile device can connect to a high security connection while changing location. In that case, the stream should not be interrupted. Hence, the system requires more flexible stream packets. Similar to that, the receiver may suffer from mobile Internet connection coverage issues or the connection of the mobile device can be highly congested for an interval of time. In order to maintain the user satisfaction, the stream should continue.

Regarding these aspects, HTTP streaming can be a good candidate for the desired end-to-end system. However, the delays introduced by TCP can be thought-provoking. According to recent works, the TCP delays do not introduce a worthwhile load to the system thanks to the increased bandwidth and advanced network structures [39, 8].

Because the system does not utilize any 3D or stereo based approach in the transmission step, the required network structure is as the same as other networking structures for ordinary data. The importance of the network structure in a full-chain mobile streaming system lies on the adaptivity and the flexibility as mentioned above.

By looking at the given information, HTTP based video streaming looks much more suitable for the desired end-to-end real time mobile system due to its flexibility in different firewall and NAT structures and adaptivity against changing bandwidth conditions.

CHAPTER 5

REAL TIME STEREO-VIDEO STREAMING

Real Time Live Stereo-Video Streaming is an active research area. Both stereo video streaming and real time video streaming are not new technologies. Nevertheless, development of displaying technologies and increasing bandwidth of the Internet connection with the recent improvements on network physical layers such as fiber-optics and wireless technologies change the course of 3D transmission and displaying technologies. In this chapter, the proposed real time stereo-video streaming system is explained.

5.1 Related Work

There are different works on real time video streaming and live video streaming in the literature. These works mostly use encoding systems based on the state-of-the-art encoding standard H.264/AVC. As transmission protocol for these video streaming systems, RTP/RTCP/UDP/IP, DVB-H, UDP Unicast, and HTTP protocols are used. As displaying method, autostereoscopic 3D displays are the mostly used displays. In addition to that, 3D displays with glasses and projection based displays are also used.

5.1.1 3DTV and Mobile 3DTV Projects

3DTV and Mobile 3DTV projects were supported by European Union Framework Programmes for Research and Technological Development. 3DTV is a project which aims to coordinate and integrate the 3D research among the researchers in Europe [59]. It is a 6^{th} Framework (FP6) project and a new conference named 3DTV-Con is started by the participants of this project. Most of the researches done on 3DTV project created a basis for studies on 3D technologies and various new projects such as Mobile 3DTV project.

Mobile 3DTV was a project about the end-to-end whole system chain from creation of 3D content to the delivery to mobile devices and is an EU 7^{th} Framework project. A new mobile autostereoscopic 3D display and the whole end-to-end streaming chain has been designed and produced as the outcome of this project. Mobile 3DTV project consists of different work areas such as stereo video content creation and coding, error resilient transmission, visual quality optimization, portable terminal device development, and execution of subjective quality tests [53].

There are two works which can be counted as end-to-end systems in these projects. Gotchev

et al. presented an end-to-end-system which utilizes H.264 simulcast encoding side-by-side image representation, RTP based transmission on DVB-H standard and autostereoscopic 3D display with OMAP 3430 based displaying system [30]. Bici *et al.* proposed an encoding and transmission system based on H.264/AVC simulcast and MVC encoding with both same bitrate and same quality, and RT based transmission on DVB-H [10]. This work does not propose any displaying system, nonetheless, appropriate 3D displaying systems can be used to complete the full chain.

Both of these projects were focused on the broadcast of the 3D content instead of unicast or multicast of the created streams. Encoding methods employed in both projects are mainly based on H.264/AVC and H.264/MVC. Streaming is based on RTP/RTCP and as the physical layer DVB-H is used. As displaying systems, different 3D displaying systems are used such as polarized projectors, autostereoscopic, volumetric and holographic displays. With these contributions, these projects cover some of the motivational reasons as mobile system. However, they are using a digital frequency modulation based broadcast method which can be disadvantageous for long distances and not very easy to implement on the transmitter side as they need special equipments for streaming.

5.1.2 Internet based Real-Time 3D Streaming

Other than broadcast based 3DTV studies, there are different works based on streaming over the Internet. These works differ in many ways. The acquisition is generally made by either stereo cameras or camera arrays [68, 49]. Aksay *et al.* and Willner *et al.* use H.264/AVC and H.264/MVC based encoding algorithms [4, 97], whereas others use different custom encoding methods [68, 100]. For the transmission part, different methods are used such as RTP/UDP/IP, HTTP, and UDP unicast [68, 40, 4, 100].

As end-to-end systems, two systems step forward. Aksay *et al.* has proposed an end-to-end 3D video straming system which uses stereo modified H.264/AVC method for encoding, RTP/UDP/IP structure for streaming and different 3D display systems such as autostereoscopic 3D, polarized projection and monocular displaying methods [4, 60]. As another study of an end-to-end system, Lamboray *et al.* created a system with point based image representations of the acquired image and a different method of encoding of point cloud data structure based on an MPEG frame structure. The streaming of the encoded point cloud bitstreams are carried out by a custom communication framework based on RTP/RTCP streaming method, and decoded points clouds are stored as JPEG images [40]. These systems offers solutions to some of motivational reasons as internet based streamer. Nevertheless, implementation of these systems are not carried out on the mobile systems, and the communication protocols they use may require a specialized network structure with some permissions. Hence, the implementation of these systems on portable platforms may be troublesome.

5.2 System Overview

The proposed system includes different components such as capture of 3D stereo video, processing, compression, transmission, streaming and display parts as it is shown in Figure 5.1. This end-to-end system consists of many parts and they have to be analyzed in different sections. The following section analyze each part of this end-to-end system and gives the

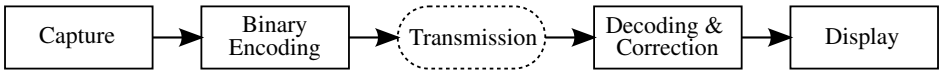


Figure 5.1: System Overview

analysis of experimental results or comparison if there is any. Most of these parts are located in the transmitter part whereas some of them are on the receiver side. Quality perception issues at the user-end are also discussed in this chapter.

5.3 Transmitter Side

In most of the video transmission systems, the main workload is on the transmitter/server side. The capture of stereo video, processing, compression, transmission and streaming stages of this end-to-end system are completed in the transmitter side. There are different software packages for this transmitter side in which all the aspects are included. In this thesis, all of these aspects are analyzed part by part and are realized by using different open-source based codes.

5.3.1 Hardware

In the transmitter stage, a low power mobile device is used. In order to utilize an embedded device, several embedded devices are researched. Different embedded platforms are appropriate for embedded multimedia streaming such as BeagleBoard, BeagleBoard-xM, Panda Board, Rapsberry Pi. The transmitter is chosen as a BeagleBoard-xM which has a 1-GHz ARM® Cortex^{TM}-A8 processor and a 800 MHz C64x DSP core. In this system, only ARM core of the card is utilized in order to show the ability to be implemented on other embedded devices without hardware codecs. BeagleBoard-xM has 512 MB DDRRAM and Ubuntu 12.04 LTS for ARM is installed on the device as operating system. A snapshot of the board can be seen in Figure 5.2. A detailed explanation of the trasmitter platform is given in Appendix B.

This embedded platform has different auxiliary units in order to help the whole process. These units have different purposes. For input and output devices, a mouse, keyboard and an LCD screen is used. For capturing images, two USB web-cameras are used, and the placement of these cameras is arranged as in the stereo pair formation which simulates the horizontal separation between human eyes. Also, both cameras are arranged to simulate the human eyes.

5.3.2 Capture and Processing

Capturing of the images is conducted by utilizing two USB web-cameras. By connecting the USB cameras to the BeagleBoard-xM via the USB ports, cameras can be used for image capture process. In order to capture images, a C++ code is written based on Video for Linux (v4l) library of the Linux operating system. This program drives both cameras and takes

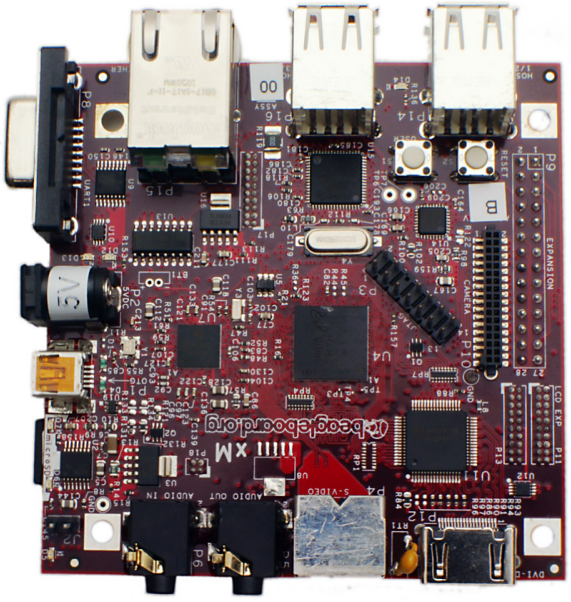


Figure 5.2: A Picture of the BeagleBoard-xM from Top

the images in YUV4:2:0 sampling structure. In order to use this program, the following command should be entered to the console.

```
$ ./luv -s 2 -c 100000
```

After the acquisition of the frames, both frames are stitched together in side-by-side formation. The code snippet which is used for both capturing and processing the obtained frames is given in Appendix B.

5.3.3 Compression

In order to lower the bitrate and send more information over the Internet, the acquired video frames must be compressed. After the stitching, the frames are merged into one frame and the resulting frame is compressed by using the H.264/AVC Stereo SEI Message encoding method. This method is chosen for the delivery, because there are not any open-source multiview MVC encoding and decoding software on the receiver side device which uses Android OS.

The main purpose is to use an efficient and back-compatible compression standard. The H.264/AVC is a commonly used video compression standard. For H.264/AVC there are many open-source encoders and decoders on both transmitter and receiver platforms. The compression is done using the x264 module of the VLC Media Player. The parameters used in encoding is taken from the experimental results as they are discussed in the Section 3.5. These parameters are given in the x264 CLI command which is provided in the same section and repeated here for reader's convenience:

```
$ x264 --profile main --keyint 8 --bframes 0 --qp 26 --frame 752 --psnr --me
```

```
dia --subme 1 --no-chroma-me --merange 32 --ref 4 --deblock 0:0 --weightp 0
--no-weightb --no-cabac --no-progress --output [outputFile] [inputFile]
[resolution]
```

In order to set the compression options following parameters are used:

- **Profile:** *Profile* gives the information to the encoder about the tool-chain and the parameter range to be used in the encoding process. The main profile has been chosen in the compression part due to its utilization of B-frames (Bi-predictive pictures).
- **Keyint:** *Keyint* parameter sets the maximum number of frames before an Instantaneous Decoder Refresh (IDR) frame (or I-frame). IDR frames restart the referencing structure of the encoder by resetting the buffer used for referencing.
- **Bframes:** *Bframes* parameter gives the maximum number of B-frame occurrences. This number is chosen equal to *keyint* for a full hierarchical encoding structure and is chosen zero for an IPP encoding structure.
- **Quantization Parameter:** Quantization parameter sets the frequency, and also implicitly the levels of quantization, of the quantization of the resulting DCT coefficients. The higher this parameter becomes, the higher the resulting frame bitrate decreases.
- **Me, Merange and Subme:** These three parameters are for choosing the motion estimation method, range and the subpixel estimation complexity. In order to increase the bitrate, these values are optimized and set for the most time-efficient values.

In the transmitter side, the *cvlc* is used for encoding which uses an x264 encoder plug-in. The input is acquired by piping the previous step's output as in YUV 4:2:0 format. The encoding parameters are entered as they are given in x264. Resulting command for the console is given:

```
$ ./luv -s 2 -c 100000 | cvlc - -vvv --demux rawvideo --rawvid fps 10
  --rawvid-width 320 --rawvid-height 120 --rawvid-chroma i420 --sout
  '#transcode{vcodec=h264,acodec=none,vb=300,venc=x264{profile=main,
  keyint=8, bframes=8, qp=26, b-adapt=0, bpyramid, no-cabac,no-weightb,
  ref=3} ,width=320, height=120, fps=10, ab=5, deinterlace}'
```

5.3.4 Streaming

As discussed in Chapter 4, there are different video streaming methods in the literature. However, the most effective one is to use packet-switched Internet streaming in a wireless channel medium for a mobile transmitter platform. HTTP video streaming on the TCP transport channel is utilized in the video streaming part due to different advantages such as the active communication channel for feedback and the adaptive transmission characteristics.

In order to stream the compressed video, VLC Media Player, an open-source software distributed by Videolan community [94], is used as streaming server. The parameters of the streaming are as follows:

```
" ... :http{mux=ffmpeg{mux=flv},dst=:8080/} "
```

After that addition to the *cvlc* command, the final form of the required command which will be entered to the console becomes as follows:

```
$ ./luv -s 2 -c 100000 | cvlc - -vvv --demux rawvideo --rawvid fps 10
  --rawvid-width 320 --rawvid-height 120 --rawvid-chroma i420 --sout
  '#transcode{vcodec=h264,acodec=none,vb=300,venc=x264{profile=main,
  keyint=8, bframes=8, qp=26, b-adapt=0, bpyramid, no-cabac,no-weightb,
  ref=3} ,width=320, height=120, fps=10, ab=5, deinterlace}:http{mux=
  ffmpeg{mux=flv},dst=:8080/}'
```

For that, the RTSP, RTP and HTTP video streaming protocols are compared in a full chain test. The most convenient video streaming protocol is determined as HTTP. There are different reasons for that such as keeping an active TCP connection between the devices in order to understand the transmission network characteristics. These characteristics are very important for analyzing the channel behavior and the perceived video quality level can be estimated by using the network characteristics. By reaching the network parameters, the video streaming can become an adaptive process and the whole system obtains more efficient state.

5.4 Receiver Side

In a mobile real-time stereo video streaming system, the receiver needs to be an efficient, powerful and mobile device. A device running Android OS with high processing capabilities is used on the receiver side. HTC Evo 3D smart-phone, the mobile receiver platform, has different processing capabilities such as the 1.2 GHz Snapdragon S3 Dual Core, 1 GB RAM, 4 GB eMMC storage, Qualcomm Adreno 220 GPU and switchable autostereoscopic display utilizing parallax barrier. Different sensors such as accelerometer, gyroscope, digital compass, proximity sensor, and ambient light detector that can aid the user experience and media delivery methods also exist on the receiver platform. More detailed explanations are given in Appendix C. The autostereoscopic screen of HTC Evo 3D allows 3D content delivery from a light, mobile and powerful device to the end-users with high quality.

5.4.1 Rendering and Display

In order to deliver the 3D content to the user, some rendering of the acquired special video stream has to be conducted. HTC Evo 3D is able to render stored compatible video files with the frame-packing information on metadata SEI is set as one of the side-by-side, top-bottom, or interlaced. However, those videos must be stored either on the SD card of the smart-phone or on the temporary memory in a progressive download from a distant stored location. There is not any stereo media player capable of 3D rendering for live streaming videos. In order to complete the system chain, the rendering of the acquired stream must be done. For that, an open-source media player, VLC Media Player for Android, has been modified and installed on the receiver platform.

For the rendering part, HTC Open Sense SDK has been utilized for its Stereoscopic 3D API [33]. In order to utilize the rendering by using the Stereoscopic API, the following steps must be done on the given order:

- A new project should be started or imported on any Android development environment
- The JAR library of the HTC Open Sense SDK must be imported into the Android project
- A code snippet with *enableS3D(true, holder.getSurface())* should be added in the source code (see Appendix D for detailed explanations)
- The project should be compiled and installed on the device

After the modification on the VLC Media Player for Android, the player works for only given (i.e. Side-by-Side) picture format and converts any frame into 3D assuming its frame-packing format is side-by-side. Hence, receiver device HTC Evo 3D starts rendering upon start and the parallax barrier between the LCD and the observer is activated.

5.4.2 Perceived Quality

For any video transmission and video streaming application, the perceived video quality is an important factor. However, there is not any way for any computer to calculate the perceived quality by a human observer. The quality perception is a very subjective issue and it depends on the content, the artifacts of the stream, visual quantitative parameters, different attributes of the observer such as the age, gender, socio-economical differences, previous experiences and even mood.

The quantitative parameters can be experimented and an estimation can be made depending on these numerical values. There are different methods to estimate the quality depending on the measurable parameters of the video stream chain, also called Quality of Service (QoS) [62]. However, other parameters introduce a great amount of uncertainty on the quality due to its natural structure of user-centric experience. This different quality estimation works are called as Quality of Experience (QoE) as the main goal of these studies to find a relation between the quality score and the human experience. There are different studies on QoS - QoE mapping [35, 57, 81, 75]; however, this area is still open to the research.

In addition to the end-to-end real time video streaming chain design, the quality perception issues are another focus of research beside this thesis work. Within the Multimedia Research Group of Middle East Technical University, there is a study on perceived quality which exploits different quantitative parameters in order to estimate the QoE carried out [102]. This study utilizes the HVS response to the moving frames and the network losses by taking the zero-motion vector ratio to bitrate and maps the obtained scores to a function according to the network losses by considering the training results. This work can estimate the perceived quality similar to the human observers' perception, and can be implemented on this end-to-end system after the addition of the depth perception criteria as a future work.

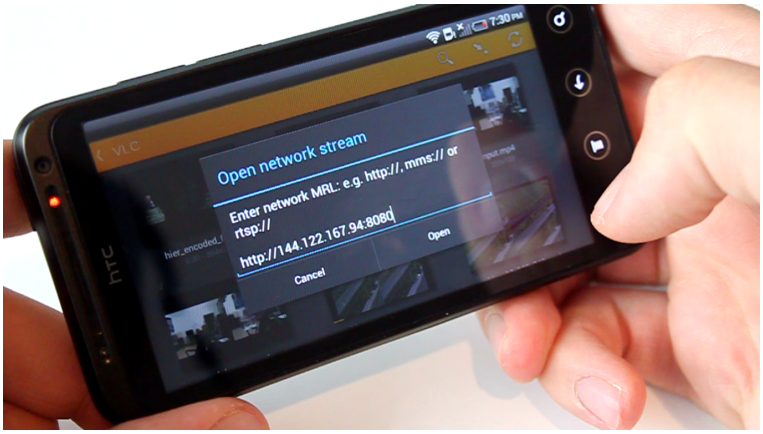


Figure 5.3: Entering the Path to the Transmitted Stream

5.5 General Operation of the Proposed System

The proposed end-to-end system consists of different parts such as the capture, processing, compression, transmission, and display. Being mobile on both transmitter and receiver sides, the system has a great advantage in portable applications. System requires a power source with 5V and 750 mA and an additional power need for input/output equipments, mainly for display. Both transmitter and receiver needs internet connection from a wireless access point.

The general operation of this system is as follows. The image acquisition and transmitting starts with the following command after the BeagleBoard-xM has plugged and the operating system has been started:

```
$ ifconfig      # This is required to find out the IP address
$ ./luv -s 2 -c 100000 | cvlc - -vvv --demux rawvideo --rawvid fps 10
  --rawvid-width 320 --rawvid-height 120 --rawvid-chroma i420 --sout
  '#transcode{vcodec=h264,acodec=none,vb=300,venc=x264{profile=main,
  keyint=8, bframes=8, qp=26, b-adapt=0, bpyramid, no-cabac,no-weightb,
  ref=3} ,width=320, height=120, fps=10, ab=5, deinterlace}:http{mux=
  ffmpeg{mux=flv},dst=:8080/}'
```

After finding the IP address and starting the streaming, the stream can be accessed by following the connection of *http://<IPaddressFound>:8080/*. The path of connection should be entered into the receiver device media player as it is shown in Figure 5.3. After entering path to the receiver media player, the stream starts to play after a brief time of establishing connection and buffering. The tests showed that the connection continues to stream and play more than 15 minutes in the trial conducted in the Multimedia Research Group Laboratory in Middle East Technical University on wireless internet connection (IEEE 108.11b/g/n) on both devices.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

6.1 Summary

Throughout this thesis, various parts of an end-to-end multimedia streaming system has been analyzed and discussed. These parts can be named as capture, encoding, transmission, display, and user satisfaction. Each different aspect of the system requires distinctive knowledge of computer engineering and signal processing topics.

In 3D multimedia systems, capturing the scene and representing the rendered image is a delicate issue. For the minimal disturbance to the human eyes, the human 3D depth perception is discussed and the differences between several 3D image representation techniques and displaying methods have been visited.

There are different 3D encoding methods proposed in the literature. Multimedia encoding by using an embedded low power device can be problematic due to the inferior computing power of embedded platform. In order to find the most efficient encoding method, experiments are conducted on both a desktop computer and a mobile platform.

Transmission of the encoded bitstream is the most vulnerable part of an end-to-end video streaming system. In order to make this step credible, several transmission methods are analyzed and compared in terms of the ability to multicast, stream real time video, adapt to network conditions, and supply QoS parameters.

On the displaying part, an open-source Android software on the receiver platform is modified for the autostereoscopic display of the 3D content to the human observer. In this part, quality perception is an important concept for the 3D perception. Even though the quality estimation is not implemented in this work, a study towards a 3D video quality estimation algorithm has began.

6.2 Conclusions and Future Works

In this thesis, a real time stereo-video streaming system from embedded platforms to mobile devices has been proposed. After analyzing various sides of an end-to-end video streaming system, the results showed that each side has different needs for different conditions. When utilizing an embedded and portable system as transmitter, the need for an efficient and low cost encoding algorithm becomes a very important issue. The experimental results

showed that the hierarchical encoding structure of H.264/MVC and hierarchical encoding structure of H.264/AVC Simulcast with side-by-side image representation methods have the best quality for the minimum bitrate. The encoding process becomes the weakest link when it is looked at the efficiency and complexity trade-off. In order to find the optimum point of this trade-off, sub-optimal solutions are used for efficiency.

After comparing several streaming protocols, the RTSP and HTTP proved to be the two commonly used adaptive streaming protocols. RTSP is developed solely for multimedia streaming purposes. On the other hand, HTTP streaming can traverse through the whole existing nodes of the Internet, can by-pass NAT and firewalls. In addition to these advantages, adaptive HTTP streaming is commonly used by many others as a result of decreasing the importance of TCP delays and increasing applicability of the TCP back-channel structure.

Compared to other related systems, the proposed system has three advantages: easy implementation, large coverage area by the utilization of the Internet, and portability. The system can be implemented easily by acquiring a cheap hardware and installation. The wireless internet usage provides both portability and large coverage area compared to digital frequency modulation based DVB-H systems.

There are some points that can be improved in the proposed system. Currently, the DSP core of the BeagleBoard-xM is not utilized due to the the conceptual compliance of the system with non-DSP embedded systems. In order to increase the performance of this system, the hardware encoder can be utilized in the DSP core of the DM 3730 System on Chip on the BeagleBoard-xM.

As indicated before, the perceived video quality by the end user is an important factor in all multimedia streaming systems. Development of a fully functional 3D video quality estimation metric can be a good direction for the future research.

REFERENCES

[1] T. Ahmed, A. Mehaoua, R. Boutaba, and Y. Iraqi. Adaptive packet video streaming over IP networks: a cross-layer approach. *Selected Areas in Communications, IEEE Journal on*, 23(2):385–401, 2005.

[2] G. Akar, A. Tekalp, C. Fehn, and M. Civanlar. Transport methods in 3DTV - a survey. *IEEE Transactions on Circuits and Systems for Video Technology (CSVT)*, 17(11):1622–1630, 2007.

[3] G. B. Akar, M. O. Bici, A. Aksay, A. Tikanmäki, and A. Gotchev. Mobile stereo video broadcast. *Mobile3DTV Project report, available online*, 2008.

[4] A. Aksay, S. Pehlivan, E. Kurutepe, C. Bilen, T. Ozcelebi, G. B. Akar, M. R. Civanlar, and A. M. Tekalp. End-to-end stereoscopic video streaming with content-adaptive rate and format control. *Signal Processing: Image Communication*, 22(2):157–168, 2007.

[5] A. A. Alatan, Y. Yemez, U. Gudukbay, X. Zabulis, K. Muller, Ç. E. Erdem, C. Weigel, and A. Smolic. Scene representation technologies for 3DTV—a survey. *Circuits and Systems for Video Technology, IEEE Transactions on*, 17(11):1587–1605, 2007.

[6] J. G. Apostolopoulos, W.-t. Tan, and S. J. Wee. Video streaming: Concepts, algorithms, and systems. *HP Laboratories, report HPL-2002-260*, 2002.

[7] t. BeagleBoard Community. BeagleBoard-xM. in beagleboard.org, `http://beagleboard.org/Products/BeagleBoard-xM`, 2013. [Online; accessed 16-August-2013].

[8] A. Begen, T. Akgul, and M. Baugher. Watching video over the web: Part 1: Streaming protocols. *Internet Computing, IEEE*, 15(2):54–63, 2011.

[9] J. Belton. Digital 3D cinema: Digital cinema's missing novelty phase. *Film History: An International Journal*, 24(2):187–195, 2012.

[10] M. Bici, D. Bugdayci, G. Akar, and A. Gotchev. Mobile 3D video broadcast. In *IEEE International Conference on Image Processing (ICIP)*, pages 2397–2400, 2010.

[11] A. Boev, M. Poikela, A. Gotchev, and A. Aksay. Modelling of the stereoscopic HVS. Technical report, Mobile3DTV Project, 2009.

[12] M. F. Bradshaw and B. J. Rogers. The interaction of binocular disparity and motion parallax in the computation of depth. *Vision Research*, 36(21):3457–3468, 1996.

[13] D. Brewster. *The stereoscope: Its history, theory and construction*. 1856.

[14] H. Brust, A. Smolic, K. Mueller, G. Tech, and T. Wiegand. Mixed resolution coding of stereoscopic video for mobile devices. In *3DTV Conference: The True Vision-Capture, Transmission and Display of 3D Video, 2009*, pages 1–4. IEEE, 2009.

[15] D. Bugdayci. Stereo video broadcasting over DVB-H. Master's thesis, Middle East Technical University, 2012.

[16] C. Cigla and A. Alatan. An efficient hole filling for depth image based rendering. In *Multimedia and Expo Workshops (ICMEW), IEEE International Conference on*, 2013.

[17] G. Coley. BeagleBoard-xM Rev C2 system reference manual. in circiutco and github, `https://github.com/CircuitCo/BeagleBoard-xM-RevC2/blob/master/Beagle_SRM_XM_C2_0_0.pdf?raw=true`, October 2012. [Online; accessed 16-August-2013].

[18] D. Digital Video Broadcasting Project. About DVB.

[19] N. A. Dodgson. Autostereoscopic 3D displays. *Computer*, 38(8):31–36, 2005.

[20] I. Draft. Recommendation and final draft international standard of joint video specification (itu-t rec. h. 264| iso/iec 14496-10 avc). *Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, JVTG050*, 2003.

[21] T. ETSI. TR 102 377 v1. 4.1 "implementation guidelines for DVB handheld services". MOBILE3DTV Project.

[22] G. E. Favalora. Volumetric 3D displays and application infrastructure. *Computer*, 38(8):37–44, 2005.

[23] C. Fehn. *3D TV Broadcasting*, pages 23–38. John Wiley & Sons, Ltd, 2006.

[24] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext transfer protocol–HTTP/1.1. in IETF, `http://tools.ietf.org/html/rfc2616`, 1999. [Online; accessed 14-August-2013].

[25] K. Fliegel. Advances in 3D imaging systems: Are you ready to buy a new 3D TV set? In *International Conference on Radioelektronika (RADIOELEKTRONIKA)*, pages 1–6, 2010.

[26] R. Frederick and V. Jacobson. RFC 3550 - RTP: A transport protocol for real-time applications, 2003.

[27] Google. The mobile movement: Understanding smartphone users. in Think Insights, `http://www.google.com/think/research-studies/the-mobile-movement.html`, April 2011. [Online; accessed 12-August-2013].

[28] A. Gotchev, G. Akar, T. Capin, D. Strohmeier, and A. Boev. Three-dimensional media for mobile devices. *Proceedings of the IEEE*, 99(4):708–741, 2011.

[29] A. Gotchev, S. Stankovic, D. Strohmeier, D. Bugdayci, G. Bozdagi, H. Akar, and N. Vladimirov. Complete end-to-end 3DTV system over DVB-H. MOBILE3DTV Project.

[30] A. Gotchev, A. Tikanmaki, A. Boev, K. Egiazarian, I. Pushkarov, and N. Daskalov. Mobile 3DTV technology demonstrator based on OMAP 3430. In *International Conference on Digital Signal Processing (DSP)*, pages 1–6, 2009.

[31] N. S. Holliman. 3d display systems. In J. P. Dakin and R. G. W. Brown, editors, *Handbook of Optoelectronics*. IOP Press, 2006.

[32] I. HTC. HTC EVO 3D'niz, kullanım kılavuzu. in htc.com, `http://dl4.htc.com/web_materials/Manual/HTC_EVO3D/HTC_EVO_3D_ICS_User_Guide_TRK.pdf`. [Online; accessed 20-August-2013].

[33] C. HTC Developers. OpenSense SDK. in htcdev, `https://www.htcdev.com/devcenter/opensense-sdk`, 2013. [Online; accessed 20-August-2013].

[34] J. Hunter, V. Witana, and M. Antoniades. A review of video streaming over the internet. *White paper http://www.dstc.edu.au/RDU/staff/jane-hunter/video-streaming.html*, 1997.

[35] K. Iwata, Y. Ishibashi, N. Fukushima, and S. Sugawara. QoE assessment in haptic media, sound, and video transmission: Effect of playout buffering control. *Computers in Entertainment (CIE)*, 8(2):12, 2010.

[36] K. Johansson, J. Bergman, D. Gerstenberger, M. Blomgren, and A. Wallen. Multi-carrier HSPA evolution. In *IEEE Vehicular Technology Conference*, pages 1–5, 2009.

[37] A. Jones, I. McDowall, H. Yamada, M. Bolas, and P. Debevec. Rendering for an interactive 360 light field display. In *ACM Transactions on Graphics (TOG)*, volume 26, page 40. ACM, 2007.

[38] T. Kawai. 3D displays and applications. *Displays*, 23(1–2):49 – 56, 2002.

[39] R. Kuschnig, I. Kofler, and H. Hellwagner. An evaluation of TCP-based rate-control algorithms for adaptive internet streaming of H. 264/SVC. In *Proceedings of the first annual ACM SIGMM conference on Multimedia systems*, pages 157–168. ACM, 2010.

[40] E. Lamboray, S. Wurmlin, and M. Gross. Real-time streaming of point-based 3D video. In *Virtual Reality, 2004. Proceedings. IEEE*, pages 91–281. IEEE, 2004.

[41] J. Y. Lee, H.-C. Wey, and D.-S. Park. A fast and efficient multi-view depth image coding method based on temporal and inter-view correlations of texture images. *Circuits and Systems for Video Technology, IEEE Transactions on*, 21(12):1859–1868, 2011.

[42] S. Lee, S. Lee, B. Oh, K.-J. Oh, I. Lim, J. Y. Lee, and C. Kim. 3d video format and compression methods for efficient multiview video transfer. In *Consumer Communications and Networking Conference (CCNC), 2012 IEEE*, pages 10–14. IEEE, 2012.

[43] Y. Liu, S. Ci, H. Tang, and Y. Ye. Application-adapted mobile 3D video coding and streaming—a survey. *3D Research*, 3(1):1–6, 2012.

[44] M. Luby, T. Stockhammer, and M. Watson. IPTV systems, standards and architectures: Part ii-application layer FEC in IPTV services. *Communications Magazine, IEEE*, 46(5):94–101, 2008.

[45] K. Maeno, N. Fukaya, O. Nishikawa, K. Sato, and T. Honda. Electro-holographic display using 15mega pixels LCD, 1996.

[46] A. Majumda, D. G. Sachs, I. V. Kozintsev, K. Ramchandran, and M. M. Yeung. Multicast and unicast real-time video streaming over wireless LANs. *Circuits and Systems for Video Technology, IEEE Transactions on*, 12(6):524–534, 2002.

[47] S. Mao, S. Lin, Y. Wang, S. S. Panwar, and Y. Li. Multipath video transport over ad hoc networks. *Wireless Communications, IEEE*, 12(4):42–49, 2005.

[48] Y. Mao, G. Cheung, A. Ortega, and Y. Ji. Expansion hole filling in depth-image-based rendering using graph-based interpolation. In *Acoustics, Speech and Signal Processing, IEEE International Conference on*, Vancouver, Canada.

[49] W. Matusik and H. Pfister. 3D TV: a scalable system for real-time acquisition, transmission, and autostereoscopic display of dynamic scenes. *ACM Transactions on Graphics (TOG)*, 23(3):814–824, 2004.

[50] P. Merkle, H. Brust, K. Dix, K. Muller, and T. Wiegand. Stereo video compression for mobile 3D services. In *3DTV Conference: The True Vision-Capture, Transmission and Display of 3D Video, 2009*, pages 1–4. IEEE, 2009.

[51] P. Merkle, A. Smolic, K. Muller, and T. Wiegand. Multi-view video plus depth representation and coding. In *Image Processing, 2007. ICIP 2007. IEEE International Conference on*, volume 1, pages I–201. IEEE, 2007.

[52] S. Misra, M. Reisslein, and G. Xue. A survey of multimedia streaming in wireless sensor networks. *Communications Surveys & Tutorials, IEEE*, 10(4):18–39, 2008.

[53] Mobile3DTV. Mobile 3DTV content delivery optimization over DVB-H system, first public summary. *Mobile3DTV Project summary, available online*, April 2009.

[54] Mobile3DTV. Mobile 3DTV content delivery optimization over DVB-H system, final public summary. *Mobile3DTV Project summary, available online*, March 2011.

[55] K. Müller, H. Schwarz, D. Marpe, C. Bartnik, S. Bosse, H. Brust, T. Hinz, H. Lakshman, P. Merkle, H. Rhee, et al. 3D high efficiency video coding for multi-view video and depth data. 2013.

[56] M. Nawrot. Depth from motion parallax scales with eye movement gain. *Journal of Vision*, 3(11), 2003.

[57] T. Nunome and S. Tasaka. QoE enhancement of audio-video IP transmission in cross-layer designed ad hoc networks.

[58] S. Ohtsuka and S. 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*, pages 72–77. IEEE, 1994.

[59] L. Onural, T. Sikora, J. Ostermann, A. Smolic, M. R. Civanlar, and J. Watson. An assessment of 3DTV technologies. In *NAB Broadcast Engineering Conference*, pages 456–467, 2006.

[60] S. Pehlivan, A. Aksay, C. Bilen, G. B. Akar, and M. R. Civanlar. End-to-end stereoscopic video streaming system. In *Multimedia and Expo, 2006 IEEE International Conference on*, pages 2169–2172. IEEE, 2006.

[61] Philips. 3D interface specifications. in Philips.com, `http://www.business-sites.philips.com/shared/assets/global/Downloadablefile/Philips-3D-Interface-White-Paper-13725.pdf`, February 2008. [Online; accessed 20-August-2013].

[62] I. Rec. G. 805,". *Generic functional architecture of transport networks*, 2000.

[63] S. Reichelt, R. Häussler, G. Fütterer, and N. Leister. Depth cues in human visual perception and their realization in 3d displays. In *SPIE Defense, Security, and Sensing*, pages 76900B–76900B. International Society for Optics and Photonics, 2010.

[64] T. Replay. freed free dimensional video.

[65] L. Rubio Romero. A dynamic adaptive HTTP streaming video service for google android. Master's thesis, KTH, 2011.

[66] O. Schreer, P. Kauff, and T. Sikora. *3D videocommunication*. Wiley Online Library, 2005.

[67] H. Schulzrinne. RFC 1889 - RTP: A transport protocol for real-time applications, 1996.

[68] S. Shi, W. J. Jeon, K. Nahrstedt, and R. H. Campbell. Real-time remote rendering of 3D video for mobile devices. In *Proceedings of the 17th ACM international conference on Multimedia*, pages 391–400. ACM, 2009.

[69] T. Silverston and O. Fourmaux. P2P IPTV measurement: a comparison study. *arXiv preprint cs/0610133*, 2006.

[70] A. Smith. Smartphone ownership – 2013 update. in PewResearchCenter, `http://www.pewinternet.org/~/media/Files/Reports/2013/PIP_Smartphone_adoption_2013.pdf`, June 2013. [Online; accessed 15-August-2013].

[71] A. Smolic, K. Mueller, P. Merkle, P. Kauff, and T. Wiegand. An overview of available and emerging 3D video formats and depth enhanced stereo as efficient generic solution. In *Picture Coding Symposium, 2009. PCS 2009*, pages 1–4. IEEE, 2009.

[72] A. Smolic, K. Mueller, N. Stefanoski, J. Ostermann, A. Gotchev, G. Akar, G. Triantafyllidis, and A. Koz. Coding algorithms for 3DTV: A survey. *Circuits and Systems for Video Technology, IEEE Transactions on*, 17(11):1606–1621, 2007.

[73] A. Smolic, K. Muller, K. Dix, P. Merkle, P. Kauff, and T. Wiegand. Intermediate view interpolation based on multiview video plus depth for advanced 3D video systems. In *Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on*, pages 2448–2451. IEEE, 2008.

[74] A. Smolic, G. Tech, and H. Brust. Report on generation of stereo video data base. Technical report, Mobile3DTV technical report, 2010.

[75] D. Soldani. Bridging QoE and QoS for mobile broadband networks. In *ETSI workshop on QoS, QoE and User Experience focusing on speech, multimedia conference tools*, 2010.

[76] L. Stelmach, W. J. Tam, D. Meegan, and A. Vincent. Stereo image quality: effects of mixed spatio-temporal resolution. *Circuits and Systems for Video Technology, IEEE Transactions on*, 10(2):188–193, 2000.

[77] A. J. Stienstra. Technologies for DVB services on the internet. *Proceedings of the IEEE*, 94(1):228–236, 2006.

[78] T. Stockhammer. Dynamic adaptive streaming over HTTP–: standards and design principles. In *Proceedings of the second annual ACM conference on Multimedia systems*, pages 133–144. ACM, 2011.

[79] A. Sullivan. Depthcube solid-state 3D volumetric display. In *Electronic Imaging 2004*, pages 279–284. International Society for Optics and Photonics, 2004.

[80] G. J. Sullivan. Standards-based approaches to 3D and multiview video coding. In *SPIE Optical Engineering+ Applications*, pages 74430Q–74430Q. International Society for Optics and Photonics, 2009.

[81] T. Suzuki, T. Kutsuna, and S. Tasaka. QoE estimation from MAC-level QoS in audio-video transmission with IEEE 802.11 e EDCA. In *Personal, Indoor and Mobile Radio*

Communications, 2008. PIMRC 2008. IEEE 19th International Symposium on, pages 1–6. IEEE, 2008.

[82] W.-T. Tan and A. Zakhor. Real-time internet video using error resilient scalable compression and TCP-friendly transport protocol. *Multimedia, IEEE Transactions on*, 1(2):172–186, 1999.

[83] A. S. Tanenbaum and D. J. Wetherall. *Computer networks*. Pearson Higher Ed, 2012.

[84] M. Tanimoto, M. Tehrani, T. Fujii, and T. Yendo. Free-viewpoint TV. *IEEE Signal Processing Magazine*, 28(1):67–76, 2011.

[85] G. Tech, H. Brust, K. Müller, A. Aksay, and D. Bugdayci. Development and optimization of coding algorithms for mobile 3DTV. Technical report, Tech. Rep, 2009.

[86] G. Tech, A. Smolic, H. Brust, P. Merkle, K. Dix, Y. Wang, K. Muller, and T. Wiegand. Optimization and comparision of coding algorithms for mobile 3DTV. In *3DTV Conference: The True Vision-Capture, Transmission and Display of 3D Video, 2009*, pages 1–4. IEEE, 2009.

[87] A. M. Tekalp and A. M. Tekalp. *Digital video processing*, volume 1. Prentice Hall PTR Upper Saddle river, NJ, 1995.

[88] I. The Advanced Television Systems Committee. About ATSC. in atsc.org, `http://www.atsc.org/cms/index.php/component/content/article/195`, May 2012. [Online; accessed 10-August-2013].

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

[90] B. Vandalore, W.-c. Feng, R. Jain, and S. Fahmy. A survey of application layer techniques for adaptive streaming of multimedia. *Real-Time Imaging*, 7(3):221–235, 2001.

[91] A. Vetro. Frame compatible formats for 3D video distribution. In *Image Processing (ICIP), 2010 17th IEEE International Conference on*, pages 2405–2408. IEEE, 2010.

[92] A. Vetro. Representation and coding formats for stereo and multiview video. In *Intelligent Multimedia Communication: Techniques and Applications*, pages 51–73. Springer, 2010.

[93] A. Vetro, A. M. Tourapis, K. Muller, and T. Chen. 3D-TV content storage and transmission. *Broadcasting, IEEE Transactions on*, 57(2):384–394, 2011.

[94] Videolan. VLC media player. in Videolan, `http://www.videolan.org/vlc/index.html`, 2013. [Online; accessed 16-August-2013].

[95] B. A. Wandell. Foundations of vision. 1995.

[96] Y. Wang, J. Ostermann, and Y.-Q. Zhang. *Video processing and communications*, volume 5. Prentice Hall Upper Saddle River, 2002.

[97] K. Willner, K. Ugur, M. Salmimaa, A. Hallapuro, and J. Lainema. Mobile 3D video using MVC and N800 internet tablet. In *3DTV Conference: The True Vision-Capture, Transmission and Display of 3D Video, 2008*, pages 69–72. IEEE, 2008.

[98] D. Wu, Y. T. Hou, W. Zhu, H.-J. Lee, T. Chiang, and Y.-Q. Zhang. On end-to-end transport architecture for MPEG-4 video streaming over the internet. *IEEE Trans on Circuits and Systems for Video Technology*, 10(6):923–941, 2000.

[99] Y. Xiao, X. Du, J. Zhang, F. Hu, and S. Guizani. Internet protocol television (IPTV): the killer application for the next-generation internet. *Communications Magazine, IEEE*, 45(11):126–134, 2007.

[100] B. Xin, R. Wang, Z. Wang, W. Wang, C. Gu, Q. Zheng, and W. Gao. AVS 3D video streaming system over internet. In *Signal Processing, Communication and Computing (ICSPCC), 2012 IEEE International Conference on*, pages 286–289. IEEE, 2012.

[101] T. Yendo, T. Fujii, M. Tanimoto, and M. Panahpour Tehrani. The seelinder: Cylindrical 3D display viewable from 360 degrees. *Journal of visual communication and image representation*, 21(5):586–594, 2010.

[102] E. Zerman, G. Akar, B. Konuk, and G. Nur. Spatiotemporal no-reference video quality assessment model on distortions based on encoding. In *Signal Processing and Communications Applications Conference (SIU), 2013 21st*, pages 1–4, 2013.

APPENDIX A

JOINT MULTIVIEW VIDEO CODING (JMVC) REFERENCE SOFTWARE

Joint Multiview Video Coding (JMVC) reference software is a primary multiview video encoder developed by the Fraunhofer Heinrich Hertz Institute HHI in Berlin, and is a commonly used H.264/MVC reference software in 3D Multimedia researches. The most basic difference of the H.264/MVC from the H.264/AVC is the utilization of inter-view redundancies. This software is an implementation of the most capable H.264/MVC encoder developed for testing purposes of the video encoding standard H.264/MPEG-4 AVC version 11. Hence, there are several different preferences which result in the encoding structure change.

In order to download this software, a CVS client, such as WinCVS in Windows OS, should be installed. Then, the CVS access parameters shown in table A.1 should be entered. This operation could be completed also by entering the following commands to the command line.

```
> 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
```

Downloaded software is needed to be built first. For that purpose, Microsoft Visual Studio project files are present in the main folder. Opening the project file and choosing "Build All" will create executable files in *<jmvc-folder>\bin* folder. These files include the encoder, decoder and bitstream assembler respectively:
H264AVCEncoderLibTestStatic, *H264AVCDecoderLibTestStatic*, and *MVCBitStreamAssemblerStatic*.

The JMVC software can be built on the Linux platforms as well. In order to build the

Table A.1: CVS Client Parameters for Downloading JMVC Software

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

reference software on Linux platforms, the following commands can be called:

```
$ cd JMVC/H264AVCExtension/build/linux
$ make
```

In order to use the encoder, config files are needed which includes the encoding parameters and/or options. An example of the usage of both encoder and decoder can be shown as following:

```
> H264AVCEncoderLibTestStatic.exe -vf <cfgFile> <view_id>
> H264AVCDecoderLibTestStatic <encodedBitStream> <outVid> <numOfViews>
```

The raw data obtained from the capture device (i.e. stereo or multiview camera) can be encoded as different video sequences. However, these videos should be encoded in an order from left to right depending on the referencing order. This order is necessary to maintain in order to exploit the inter-view redundancies. After the encoding MVCBitStreamAssemblerStatic executable have to be called in order to form the bitstream as time-multiplexed frame structure (i.e. V1F1-V2F1-V3F1-V1F2-...). An example of the encoding and decoding procedure of the video sequence with 3 videos (left, center, right) can be shown as following:

```
> H264AVCEncoderLibTestStatic.exe -vf encode.cfg 0
> H264AVCEncoderLibTestStatic.exe -vf encode.cfg 1
> H264AVCEncoderLibTestStatic.exe -vf encode.cfg 2
> MVCBitStreamAssemblerStatic -vf assembler.cfg

... Transmission ...

> H264AVCDecoderLibTestStatic out.264 out.yuv 3
```

There are different parameters present in the config files. These parameters change the main encoding variables such as input and output file names, the resolution of the raw data, number of frames to be encoded; and coding related parameters such as quantization parameter, GOP size etc. The full explanations of all parameters are given on the Software Manual of JMVC 8.3 which can be accessed from `http://www.eee.metu.edu.tr/~zerman/files/jmvcSoftwareManual.doc`. Sample config files for encoding and assambling are given below:

Config file for encoder:

```
# JMVC Configuration File in MVC mode

#====================== GENERAL ================================================
InputFile               input      # input file
OutputFile              stream     # bitstream file
ReconFile               rec        # reconstructed file
MotionFile              motion     # motion information file
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

#====================== INTERLACED ==========================================
MbAff                   0           # 0=frameMb, 1=MbAff
PAff                    0           # 0=frame, 1=field, 2=frame/field

#====================== STRUCTURE ===========================================
GOPSize                 12              # GOP Size (at maximum frame rate)
IntraPeriod             12   # Anchor Period
NumberReferenceFrames   2           # Number of reference pictures
InterPredPicsFirst      1           # 1 Inter Pics; 0 Inter-view Pics
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
PicOrderCntType         0           # Picture order count type (0 or 2)

#============================== 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             32          # 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)

#============================== NESTING SEI MESSAGE ==========================
NestingSEI              0           #(0: NestingSEI off, 1: NestingSEI on)
SnapShot                0           #(0: SnapShot off, 1: SnapShot on)
#========================== ACTIVE VIEW INFO SEI MESSAGE ======================
ActiveViewSEI           0           #(0: ActiveViewSEI off, 1: ActiveViewSEI on)
#====================== VIEW SCALABILITY INFOMATION SEI MESSAGE ================
ViewScalInfoSEI         0           #(0: ViewScalSEI off, 1: ViewScalSEI on)

#====================== MULTIVIEW SCENE INFORMATION SEI MESSAGE ================
MultiviewSceneInfoSEI  1 #(0: off, 1: on)
MaxDisparity 12
#=================MULTIVIEW ACQUISITION INFOMATION SEI MESSAGE ===============
MultiviewAcquisitionInfoSEI 1 #(0: off, 1: on)
AcquisitionInfoFile Camera_ballroom.cfg
```

```
#================== 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

#============== Level conformance checking of the DPB size ==============
DPBConformanceCheck     1          # (0: disable, 1: enable, 1:default)

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


NumLevelValuesSignalledMinus1 0

Level_IDC                       1
NumApplicableOpsMinus1          0
ApplicableOpTemporalId          0 0
ApplicableOpNumTargetViewsMinus1 0 2
ApplicableOpNumViewsMinus1      0 2
ApplicableOpTargetViewId        0 0 0
ApplicableOpTargetViewId        0 1 2
ApplicableOpTargetViewId        0 2 1
```

Config file for assembler:

```
#====================== Assembler: View Encode order ==========================
OutputFile              ballroom.264
NumberOfViews           8
InputFile0              stream_0.264
InputFile1              stream_2.264
InputFile2              stream_1.264
InputFile3              stream_4.264
InputFile4              stream_3.264
InputFile5              stream_6.264
InputFile6              stream_5.264
```

```
InputFile7              stream_7.264
```

APPENDIX B

TRANSMITTER PLATFORM: BEAGLEBOARD-XM

Transmitter platform in the proposed system is an open-source hardware which is promoted by Texas Instruments (TI), DigiKey and Newark element14, named as BeagleBoard-xM [7]. BeagleBoard-xM is a low power embbedded device which has 1-GHz ARM® Cortex™-A8 processor and 512 MB DDRRAM. DM 3730 System on Chip (SoC) Integrated Circuit (IC) package also have a DSP core with C64x series of TI. These properties makes BeagleBoard-xM a low-power computer. All of the hardware properties of BeagleBoard-xM is given on table B.1

BeagleBoard-xM has the ability to run well known operating systems such as Linux and Android which is built for ARM core. The operating systems that can be used in BeagleBoard-xM are mainly based on Linux, FreeBSD, OpenBSD, RISC OS, Symbian and Android. In the proposed system, the BeagleBoard-xM runs an Ubuntu 12.04 LTS distribution Linux operating system. The system has 3.2.0-23-omap #36 Linux kernel and many of programs are able to run via different package sources which compile the sources for ARM architecture.

The dimensions of BeagleBoard-xM are 82.55 by 82.55 mm. There are four USB ports, an Ethernet jack, S-Video input, DVI-D output with HDMI socket for space limitations and two jacks for audio-in and out on the board as input/output ports. The power consumption of the BeagleBoard-xM is estimated by the maximum voltage of 5 V and maximum current of 750 mA as 3.75 Watts. This low power consumption enables BeagleBoad-xM to be used in various mobile applications and platforms which can supply very low power. Used BeagleBoard-xM revision is *B*.

There are different hardware parts used in the proposed system which uses BeagleBoard-xM as the core part of the transmitter part. On the board, the capturing of the video, processing, and streaming parts are dealt with. In order to both operate the board and handle the necessary tasks, there are different input ad output devices needed such as keyboard, mouse and display. So, a mouse and a keyboard are connected to the device by USB and display device is connected to BeagleBoard-xM by a HDMI-to-DVI-D cable. The HDMI part is used only as because the socket size of HDMI is smaller than the socket size of the DVI-D. In addition to that, two webcams of A4Tech PK-636K (PK-636G can also be used) are used for image acquisition. These cameras are connected to the board by USB. The connection to the world wide web is supplied by a USB Wi-Fi dongle. The Wi-Fi dongle used in this particular system is BELKIN N150 Micro Wireless USB adapter with a very small size and with part number # F7D1102. In order to handle the number of USB connections, a USB hub is used, and aside from the Wi-Fi adapter other USB devices are connected to it.

Due to both low power consumption and high operational abilities, BeagleBoard-xM is used

Table B.1: BeagleBoard-xM Hardware Properties [17]

	Specifications
SoC Model	TI DM3730 Digital Media Procesor (Compatible with OMAP^{TM} 3 architecture)
CPU	1-GHz ARM® Cortex^{TM}-A8
GPU	Imagination Technologies PowerVR SGX^{TM} series Graphics Accelerator
DSP	TMS320C64x+^{TM} DSP core
Memory	512 MB DDR2 RAM
Debug Support	14-pin JTAG, UART, GPIO pins, 3 LEDs
Input/Output Ports	Ethernet, 4 USB, audio stereo-in and out, S-Video input, DVI-D output with HDMI socket, and SD/MMC connection
SD/MMC Connection	MicroSD card reader/writer as stored memory
Power connection	USB and DC power
Power consumption	$\tilde{3}$.75 Watts

as a portable and very-low-power computer by utilizing the Ubuntu 12.04 LTS for ARM OS. Ubuntu OS for ARM is a Linux distribution which is compiled for ARM architecture. Hence, many of the generic software applications can be run on BeagleBoard-xM. In order to manage the image capture part, a custom software is used which is based on Video for Linux (v4l) library. The main part of this software is given at the end of this appendix chapter.

B.1 Software Installation

In order to install Ubuntu 12.04 LTS on BeagleBoard-xM the following steps are completed in given order:

- Download the preinstalled Ubuntu 12.04 LTS compiled for OMAP 3x Series architecture from `http://cdimage.ubuntu.com/releases/12.04.2/release/ubuntu-12.04-preinstalled-desktop-a omap.img.gz`
- In order to clone the downloaded disk image to SD Card, *dd* command of Unix is utilized.
- Enter the following command to command line in order to clone the downloaded disk image into the bootable SD Card form:

```
$ tar -xaf ubuntu-12.04-preinstalled.img.gz | dd of=/media/SDcard
```

- Plug SD Card into SD/MMC Card reader port of BeagleBoad-xM and complete the installation by following the instructions on the screen
- In order to make system more efficient and smooth, download a different Desktop Manager by:
  - Start the terminal and run the command given below to install the Xfce a light-weight desktop manager instead of pre-installed Gnome

– Log off the user and choose the desktop manager from the login screen before entering username and password

```
$ sudo apt-get install xfce4
```

After these operations, the BeagleBoard-xM can be opened by using the Ubuntu 12.04 LTS OS and Xfce4 Desktop manager. However, there are different packages needed to be installed in order to aid the capturing, processing and transmission. These operations need different tools and programs installed.

Firstly, v4l packages must be installed in order to use the v4l library functions in the capturing operation. In order to install v4l library tools, the following line of command is needed to be entered. After installing the v4l library tools, a link is needed to be created in order for the image acquisition program to work. The needed *ln* command is given below:

```
$ sudo apt-get install v4l2loopback-dkms v4l2loopback-source v4l2ucp v4l-conf
    v4l-utils libv4l-dev
$ ln -s /usr/include/libv4l-videodev.h /usr/include/linux/videodev.h
```

For the streaming part, VLC Media Player created by Videolan Organization [94] is needed as a streamer. In order to install VLC Media Player, the following command should be called in command line:

```
$ sudo apt-get install vlc
```

B.2 Image Capture Software

In order to acquire images from the webcams, the a custom written C program is used. This C program utilizes the Video for Linux (V4L) library in order to get images by using the Linux framework. The written program then is compiled by GCC GNU C Compiler and the created binary file is used for the image acquisition. The code part of the main file of the program is given at the end of this appendix chapter. The other necessary documents such as different C files and the whole structure of this program can be reached by following `http://www.eee.metu.edu.tr/~zerman/files/luvBeagle.zip`

Image Acquisition Program Source Code: luvcview.c

```
/*******************************************************************************
#      uvcview: Sdl video Usb Video Class grabber           .         #
#This package work with the Logitech UVC based webcams with the mjpeg feature. #
#All the decoding is in user space with the embedded jpeg decoder              #
#.                                                                             #
#  Copyright (C) 2005 2006 Laurent Pinchart &&  Michel Xhaard     #
#                                                                             #
# This program is free software; you can redistribute it and/or modify        #
# it under the terms of the GNU General Public License as published by        #
# the Free Software Foundation; either version 2 of the License, or           #
```

```
# (at your option) any later version.                                   #
#                                                                       #
# This program is distributed in the hope that it will be useful,       #
# but WITHOUT ANY WARRANTY; without even the implied warranty of        #
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the         #
# GNU General Public License for more details.                          #
#                                                                       #
# You should have received a copy of the GNU General Public License     #
# along with this program; if not, write to the Free Software           #
# Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA    #
#                                                                       #
************************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/file.h>
#include <string.h>
#include <pthread.h>
#include <linux/videodev.h>
#include <sys/ioctl.h>
#include <sys/mman.h>
#include <errno.h>
#include <fcntl.h>
#include <time.h>
#include <sys/time.h>
#include <signal.h>
#include <getopt.h>
#include "v4l2uvc.h"

/* Fixed point arithmetic */
#define FIXED Sint32
#define FIXED_BITS 16
#define TO_FIXED(X) (((Sint32)(X))<<(FIXED_BITS))
#define FROM_FIXED(X) (((Sint32)(X))>>(FIXED_BITS))

#define INCPANTILT 64 // 1

int width = 160;
int height = 120;

struct vdIn *videoIn;
struct vdIn *videoIn2;

const char *videodevice = "/dev/video0";
const char *videodevice2 = "/dev/video1";

static void process_image (const void *  p,FILE* fp)
{
static unsigned char* packed_value=NULL;
static unsigned char* packed_value2=NULL;
int i=0,j=0;
    while(i < (width * height))
    {
     packed_value = ((unsigned char*)p+i*2);
     fprintf(fp,"%c",*packed_value);
     i++;
    }
```

```
    for(j=0;j<height;j=j+2)
    {
     packed_value2 = (unsigned char*)p + width*2*j;
     for(i=0;i<width/2;i++)
     {
     packed_value = packed_value2 + i*4+1;
     fprintf(fp,"%c",*packed_value);
     }
    }

    for(j=0;j<height;j=j+2)
    {
       packed_value2 = (unsigned char*)p + width*2*j;
       for(i=0;i<width/2;i++)
     {
        packed_value = packed_value2 + i*4+3;
        fprintf(fp,"%c",*packed_value);
        }
   }
}

static void process_twoimage (const void *  p,const void * pp,FILE* fp)
{
static unsigned char* packed_value=NULL;
static unsigned char* packed_value2=NULL;
int i=0,j=0;
for(j=0;j<height;j++)
{
packed_value2=((unsigned char*)p+j*2*width);
for(i=0;i<width;i++)
{
packed_value = packed_value2 + i*2;
//fprintf(fp,"%c",*packed_value);
printf("%c",*packed_value);
}
packed_value2=((unsigned char*)pp+j*2*width);
for(i=0;i<width;i++)
{
packed_value = packed_value2 + i*2;
//fprintf(fp,"%c",*packed_value);
printf("%c",*packed_value);
}
}

    for(j=0;j<height;j=j+2)
    {
     packed_value2 = (unsigned char*)p + width*2*j;
     for(i=0;i<width/2;i++)
     {
     packed_value = packed_value2 + i*4+1;
     //fprintf(fp,"%c",*packed_value);
     printf("%c",*packed_value);
     }
     packed_value2 = (unsigned char*)pp + width*2*j;
     for(i=0;i<width/2;i++)
     {
     packed_value = packed_value2 + i*4+1;
     //fprintf(fp,"%c",*packed_value);
     printf("%c",*packed_value);
```

```
     }
    }

    for(j=0;j<height;j=j+2)
    {
       packed_value2 = (unsigned char*)p + width*2*j;
       for(i=0;i<width/2;i++)
     {
        packed_value = packed_value2 + i*4+3;
        //fprintf(fp,"%c",*packed_value);
        printf("%c",*packed_value);
        }
       packed_value2 = (unsigned char*)pp + width*2*j;
       for(i=0;i<width/2;i++)
       {
       packed_value = packed_value2 + i*4+3;
       //fprintf(fp,"%c",*packed_value);
       printf("%c",*packed_value);
       }
   }
}

static void usage (FILE * fp, int argc, char ** argv)
{
        fprintf (fp,
                 "Usage: %s [options]\n\n"
                 "Options:\n"
                 "-d | --first device name     Video device name [/dev/video]\n"
                 "-y | --help            Print this message\n"
                 "-m | --second device name    Second video device name\n"
          "-s | --state  General option\n"
 "   | 0: one camera capture and saved[default]\n"
          "   | 1: two camera capture and saved in different file\n"
          "   | 2: two camera capture and saved side by side\n"
          "-w | --width  Capture width pixel number[default=320]\n"
          "-h | --height  Capture height pixel number[default=240]\n"
          "-c | --count  Capture frame count[default=100]\n",
                 argv[0]);
}

static const char short_options [] = "d:m:w:h:s:c:y";

static const struct option long_options [] = {
        { "device",     required_argument,      NULL,           'd' },
        { "help",       no_argument,            NULL,           'h' },
        { "mmap",       no_argument,            NULL,           'm' },
        { 0, 0, 0, 0 }
};


int main (int argc, char ** argv)
{
    int format = 0;
    int grabmethod = 1;
    int state=0;
    int count=100;
    int i=0;
    for (;;) {
        int index;
        int c;
```

```
        c = getopt_long (argc, argv,short_options, long_options, &index);
        if (-1 == c)
              break;
        switch (c) {
case 0: /* getopt_long() flag */
break;
case 'd':
videodevice = optarg;
break;
case 'y':
usage (stdout, argc, argv);
exit (EXIT_SUCCESS);
case 'm':
videodevice2 = optarg;
break;

case 'w':
width = atoi(optarg);
break;

case 'h':
height = atoi(optarg);
break;

case 's':
state = atoi(optarg);
break;

case 'c':
count = atoi(optarg);
break;

default:
usage (stderr, argc, argv);
exit (EXIT_FAILURE);
        }
    }
    FILE *file;
    file = fopen("/home/emin/video1.yuv", "wb");
    FILE *file2;
    file2 = fopen("/home/emin/video2.yuv", "wb");
    FILE *file3;
    file3 = fopen("/home/emin/video12.yuv", "wb");

format = V4L2_PIX_FMT_YUYV;

//printf("Size width: %d height: %d \n", width, height);
videoIn = (struct vdIn *) calloc(1, sizeof(struct vdIn));

if (init_videoIn(videoIn, (char *) videodevice, width, height,
        format,grabmethod) < 0)
exit(1);
if (state==1 || state==2)
{
videoIn2 = (struct vdIn *) calloc(1, sizeof(struct vdIn));
if (init_videoIn(videoIn2, (char *) videodevice2, width, height,
            format,grabmethod) < 0)
exit(1);
}
```

```
if(state==0)
{
for(i=0;i<count;i++)
{
uvcGrab(videoIn);
process_image(videoIn->framebuffer,file);
}
}
if(state==1)
{
for(i=0;i<count;i++)
{
uvcGrab(videoIn);
uvcGrab(videoIn2);
process_image(videoIn->framebuffer,file);
process_image(videoIn2->framebuffer,file2);
}
}
if(state==2)
{
for(i=0;i<count;i++)
{
uvcGrab(videoIn);
uvcGrab(videoIn2);
process_twoimage(videoIn->framebuffer,videoIn2->framebuffer,
                file3);
}
}

close_v4l2(videoIn);
free(videoIn);
if(state==1 || state==2)
{
close_v4l2(videoIn2);
free(videoIn2);
}
    //printf("Clean Up done Quit \n");
    return 0;
}
```

APPENDIX C

RECEIVER PLATFORM: HTC EVO 3D

Receiver platform used in the proposed system is a mobile device with switchable autostereoscopic display. The used platform is one of the commercial autostereoscopic smart phones: HTC Evo 3D. This smart phone uses Android version 4.0.3 as its operating system.

The most important aspect of this device for choosing it as the receiver platform is its switchable autostereoscopic display. The autostereoscopic display is a 3D display type which does not require user to wear glasses as it is mentioned in Chapter 2. The advantage of autostereoscopic displays, being not required to wear glasses, gives much more freedom for the user. Also, this smart phone has different features such as the accelerometer, gyroscope, ambient light sensor and different properties that can be used in both delivering quality content and estimating the quality of the audiovisual content delivered. The whole list of the properties an be seen at Table C.1.

Table C.1: HTC Evo 3D Hardware Properties [32]

	Specifications
CPU	Snapdragon S3 Dual Core 1.2 GHz
GPU	Qualcomm Adreno 220
Memory	1 GB RAM
Storage	4 GB eMMC
Reusable Storage	8 GB micro SDHC
Screen	Switchable 2D/Autostereoscopic 3D screen with 960x540 px 4.3 inc. TFT
Cameras	5M MP LED Flash rear camera, 1.3 MP front camera
Sensors	Touch screen, Accelerometer, Gyroscope, Digital Compass, Proximity Sensor, Ambient Light Detector
Dimensions and Weight	127x67x12 mms, 170 g
Operating System	Android 4.0.3
Connectivity	Triband CDMA, 802.16e WiMAX, 802.11/b/g/n WiFi, Bluetooth v3.0, HDMI

Being Android a Linux based operating system and an open source project, the application development is very easy for any developers on the world. That creates a perfect environment for academic researches on the mobile devices with different connectivity options. The ability to add, delete or edit any options on the phone gives a great flexibility.

Besides being advantageous of HTC Evo 3D on having an autostereoscopic screen, HTC also provides an Application Programming Interface (API) in order to include the 3D abilities of the phone in custom written Android programs. In order to use the API, the following lines

should be included in the code:

```
DisplaySetting.setStereoscopic3DFormat(Surface surface, int format);

// OR in a more generic way

enableS3D(boolean enable, Surface surface)

...

private void enableS3D(boolean enable, Surface surface) {
    text.setVisibility(View.INVISIBLE);
    int mode = DisplaySetting.STEREOSCOPIC_3D_FORMAT_SIDE_BY_SIDE;
    if(!enable) {
      mode = DisplaySetting.STEREOSCOPIC_3D_FORMAT_OFF;
    }
    boolean formatResult = true;
    try {
      formatResult = DisplaySetting.setStereoscopic3DFormat(surface, mode);
    } catch (NoClassDefFoundError e) {
      text.setVisibility(View.VISIBLE);
      android.util.Log.i(TAG, "class not found - S3D display not available");
    }
    if (!formatResult) {
      android.util.Log.i(TAG, "S3D format not supported");
    }
  }
```

In order to use the HTC's Stereoscopic 3D API which is in OpenSense SDK, the OpenSense SDK must be downloaded by using the Android SDK. In order to install the development environment, the following steps must be followed:

- Download and install the Dava Development Kit (JDK) for Java SE 6 from `http://oracle.com/technetwork/java/javase/downloads`
- You may proceed by following either step:
  - Download and install the SDK and ADT Bundle for Android development in Windows by the url: `http://developer.android.com/sdk/index.html`
  - Download and install Android SDK by following the url: `http://developer.android.com/sdk/index.html` and use any other Integrated Development Environment (IDE)
- Open and select platforms 15 and 16, and click download button

APPENDIX D

VLC MEDIA PLAYER ON ANDROID

For displaying the acquired 3D video content, there is a need for 3D live video decoder. However, in the Android OS, there are not such applications which are able to receive the real-time stereo video stream and decode it. In order to fix that situation, a modified software is used for receiving the encoded streams on the receiver platform of HTC Evo 3D.

In order for one to manage the creation of a stereo-video media player, there are different possibilities to experienced. The built-in media player and as well as other media players do not support the live stereo-video stream reception and decoding, even there are different media players that can decode 3D content which is stored on a drive mounted on the system, i.e. Hard Drive. Hence, an open-source media player is aimed to be modified for 3D stream reception purposes.

For that purpose, VLC Media Player created by Videolan Organization [94] is used to create a new stereo-video streaming Android application. To realize this system, the VLC media player is copied, or cloned, from the original source and compiled on the host(developer) computer by using the following command:

```
$ git clone git://git.videolan.org/vlc/vlc-android.git
```

In order to compile the system for the VLC Media Player for Android, the Native Development Kit (NDK) is needed. For that, the download of the NDK system by following the link `http://developer.android.com/tools/sdk/ndk/index.html` and the commands needed for the compilation of VLC Media Player is given below as follows:

```
$ cd <cloneDirectory>
$ export JAVA_JDK=/usr/lib/jvm/java-6-sun
$ export PATH=${JAVA_JDK}/jre/bin:${PATH}
$ export ANDROID_SDK=<android_SDK_folder>
                     (i.e. /home/emin/android_vlc/android-sdk-linux)
$ export ANDROID_NDK=<android_NDK_folder>
                     (i.e. /home/emin/android_vlc/android-ndk-r8b)
$ export PATH=${ANDROID_SDK}/platform-tools:${ANDROID_SDK}/tools:
    ${ANDROID_NDK}:${PATH}
$ export ANDROID_ABI=armeabi-v7a
$ sh compile.sh release
```

After compiling the NDK part of the VLC Media Player, the software becomes able to be compiled on the Eclipse or any similar IDE. By following the *File -> Import -> General*

-> *Existing Projects* options and choosing the project path, the existing VLC MP project becomes able to be edited. After opening the IDE, the necessary modifications habe been done as following:

```
//-----line 84
import android.widget.Spinner;
import android.widget.TextView;

import com.htc.view.DisplaySetting; //ADDED

public class VideoPlayerActivity extends Activity {

    public final static String TAG = "VLC/VideoPlayerActivity";
...
...
//-----line 98
private static final int SURFACE_BEST_FIT = 0;
    private static final int SURFACE_FIT_HORIZONTAL = 1;
    private static final int SURFACE_FIT_VERTICAL = 2;
    private static final int SURFACE_FILL = 3;
    private static final int SURFACE_16_9 = 4;
    private static final int SURFACE_4_3 = 5;
    private static final int SURFACE_ORIGINAL = 6;
    private int mCurrentSize = SURFACE_FILL; //CHANGED:SURFACE_BEST_FIT
...
...
//-----line 151
    private String[] mSubtitleTracks;

    //3D Debugging
    private boolean s3d_enable_ez; //ADDED
    private int s3d_type_ez; //ADDED

    @Override
    @TargetApi(11)
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.player);

        s3d_enable_ez = true; //INITIALIZED - s3d_enable
        s3d_type_ez = 1; // INITIALIZED - 1 sbs - 2 tb - 3 interleaved

        SharedPreferences pref = PreferenceManager.getDefaultSharedPreferences(this);
...
...
//-----line 295
@Override
    protected void onPause() {
        if(mSwitchingView) {
            super.onPause();
            return;
        }

        long time = mLibVLC.getTime();
        long length = mLibVLC.getLength();
        //remove saved position if in the last 5 seconds
        if (length - time < 5000)
            time = 0;
        else
            time -= 5000; // go back 5 seconds, to compensate loading time
```

```
        if (mLibVLC.isPlaying()) {
            mLibVLC.pause();
        }
        mSurface.setKeepScreenOn(false);

        // Save position
        if (time >= 0) {
            SharedPreferences preferences =
                getSharedPreferences(PreferencesActivity.NAME, MODE_PRIVATE);
            SharedPreferences.Editor editor = preferences.edit();
            editor.putString(PreferencesActivity.LAST_MEDIA, mLocation);
            editor.putLong(PreferencesActivity.LAST_TIME, time);
            editor.commit();
        }
        super.onPause();
        enableS3D(false, mSurfaceHolder.getSurface()); //ADDED
    }

    @Override
    protected void onDestroy() {
        unregisterReceiver(mBatteryReceiver);
        if (mLibVLC != null && !mSwitchingView) {
            mLibVLC.stop();
        }

        EventManager em = EventManager.getIntance();
        em.removeHandler(eventHandler);

        mAudioManager = null;

        if(mSwitchingView) {
            Log.d(TAG, "mLocation = \"" + mLocation + "\"");
            AudioServiceController.getInstance().showWithoutParse(mLocation);
            Intent i = new Intent(this, AudioPlayerActivity.class);
            i.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
                Intent.FLAG_ACTIVITY_MULTIPLE_TASK);
            startActivity(i);
        }
        //AudioServiceController.getInstance().unbindAudioService(this);
        super.onDestroy();
        enableS3D(false, mSurfaceHolder.getSurface()); //ADDED
    }

...
...
//-----line 849
    /**
     * attach and disattach surface to the lib
     */
    private final SurfaceHolder.Callback mSurfaceCallback = new Callback() {
        @Override
        public void surfaceChanged(SurfaceHolder holder, int format, int width,
            int height) {

         enableS3D(s3d_enable_ez, holder.getSurface()); //ADDED
         mLibVLC.attachSurface(holder.getSurface(), VideoPlayerActivity.this,
                width, height);
        }

        @Override
```

```
        public void surfaceCreated(SurfaceHolder holder) {
        }

        @Override
        public void surfaceDestroyed(SurfaceHolder holder) {
            mLibVLC.detachSurface();
            enableS3D(false, holder.getSurface()); //ADDED
        }
    };
...
...
//-----line 1054
            mTitle.setText(title);
        }
    }


    //PART ADDED
    private void enableS3D(boolean enable, Surface surface) {
        Log.i(TAG, "enableS3D(" + enable + ")");
        int mode = 0;
        switch (s3d_type_ez) {
        case 1:
            mode = DisplaySetting.STEREOSCOPIC_3D_FORMAT_SIDE_BY_SIDE;
            break;
        case 2:
         mode = DisplaySetting.STEREOSCOPIC_3D_FORMAT_TOP_BOTTOM;
            break;
        case 3:
         mode = DisplaySetting.STEREOSCOPIC_3D_FORMAT_INTERLEAVED;
            break;
        }

        if (!enable) {
            mode = DisplaySetting.STEREOSCOPIC_3D_FORMAT_OFF;
        }
        boolean formatResult = true;
        try {
            formatResult = DisplaySetting.setStereoscopic3DFormat(surface, mode);
        } catch (NoClassDefFoundError e) {
            android.util.Log.i(TAG,
                "class not found - S3D display not available");
        }
        Log.i(TAG, "return value:" + formatResult);
        if (!formatResult) {
            android.util.Log.i(TAG, "S3D format not supported");
        }
    }
    //PART ADDED ---ends
}
//-----line 1093 ----end_of_the_code
```