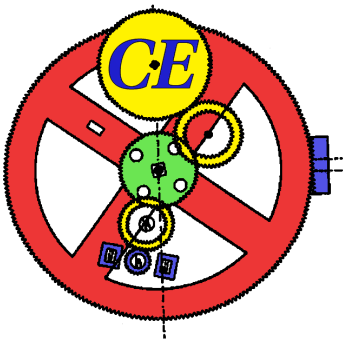


Computer Engineering
Mekelweg 4,
2628 CD Delft
The Netherlands
http://ce.et.tudelft.nl/

2010

MSc THESIS

Stereoscopic Remote Vision System

Akhil Piplani

PHILIPS

CE-MS-2010-24

Abstract

This thesis presents a remote vision system designed to be used for telepresence applications. Telepresence is essentially being able to assert one's presence at a remote location. Being able to see and talk to people in another corner of the world is one method of telepresence we have come to know as video conferencing. As technologies evolve, the sense of 'presence' allowed by them also grows. Present-day telepresence systems limit our most prominent sense, sight, to monoscopic video where we only see one view of the remote location. In order to take telepresence to the next level by being able to manipulate remote objects, it is important that our sight be stimulated to perceive depth to the fullest. Therefore, this thesis focuses on creating a *stereoscopic* remote vision system designed for teleoperation. Delay plays a major part in the ease of teleoperation. While most teleoperation systems focus of alleviating the effects of delay by means of control methodologies and environment modeling, this system attempts to directly minimize delay and other factors of human discomfort. This was done by analyzing the several technologies involved in capture, compression, transport and display of 3D video along with an exploration on the factors of human comfort and performance for teleoperation. The most appropriate technologies were then selected while keeping these factors in mind. The stereoscopic remote vision system was then designed and implemented while keeping a focus to minimize delay. The resulting system was then used as a testbench to further explore the same factors.

TU Delft
Delft University of Technology

Faculty of Electrical Engineering, Mathematics and Computer Science

Stereoscopic Remote Vision System
A Delay Minimizing Approach for Telepresence

THESIS

submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

in

EMBEDDED SYSTEMS

by

Akhil Piplani
born in New Delhi, India

Computer Engineering
Department of Electrical Engineering
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology

Stereoscopic Remote Vision System

by Akhil Piplani

Abstract

This thesis presents a remote vision system designed to be used for telepresence applications. Telepresence is essentially being able to assert one's presence at a remote location. Being able to see and talk to people in another corner of the world is one method of telepresence we have come to know as video conferencing. As technologies evolve, the sense of 'presence' allowed by them also grows. Present-day telepresence systems limit our most prominent sense, sight, to monoscopic video where we only see one view of the remote location. In order to take telepresence to the next level by being able to manipulate remote objects, it is important that our sight be stimulated to perceive depth to the fullest. Therefore, this thesis focuses on creating a *stereoscopic* remote vision system designed for teleoperation. Delay plays a major part in the ease of teleoperation. While most teleoperation systems focus of alleviating the effects of delay by means of control methodologies and environment modeling, this system attempts to directly minimize delay and other factors of human discomfort. This was done by analyzing the several technologies involved in capture, compression, transport and display of 3D video along with an exploration on the factors of human comfort and performance for teleoperation. The most appropriate technologies were then selected while keeping these factors in mind. The stereoscopic remote vision system was then designed and implemented while keeping a focus to minimize delay. The resulting system was then used as a testbench to further explore the same factors.

Laboratory : Computer Engineering
Codenumber : CE-MS-2010-24
Committee Members :

Advisor:	Sait Izmit, Philips Apptech
Advisor:	Georgi N. Gaydadjiev, CE, TU Delft
Chairperson:	Koen Bertels, CE, TU Delft
Member:	H.G. Gross, SE, TU Delft

For my family and best friend

Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank my advisors Sait Izmit and Georgi N. Gaydadjiev for their guidance, support and help. Hans Kanters for providing insightful information into the hardware aspects of this project. Last but not the least, I would like to thank my family and friends for their support.

Akhil Piplani
Delft, The Netherlands
September 7, 2010

1 Introduction

"It was only viewing, you see" – Gladia in "The Naked Sun" by Isaac Asimov.

In the science fiction novel "The Naked Sun"[4], Isaac Asimov described a futuristic society where people almost never saw each other, rather *'viewed'* each other remotely through holographic *'trimentionals'*. This novel was first published in 1957. In the present day, new *viewing* technologies are making the world smaller, making seeing each other redundant. As can be imagined, a system capable of transporting and displaying immersive video can have numerous applications. Like many technologies envisioned in science fiction, *telepresence*[39] has also found its way into real life. The term telepresence, coined in 1980 by Marvin Minsky, has come to be used widely in the context of technologies that enable people to assert their presence at a remote location.

Telepresence doesn't just consist of being able to 'view' each other. Rather, it has come to encompass situations with varied degrees of 'presence'. At the moment, academics and corporations alike have taken numerous initiatives that fall under the scope of telepresence. For example, video-conferencing rooms where several people can take part from around the globe, using large screens and microphones while providing guaranteed quality-of-service; defense personnel using robots for diffusing bombs or for surveillance. Further, there is ongoing research where the fields of telepresence and augmented reality are combined. As we achieve more natural telepresence capabilities, the applications become more immersive, intuitive and most importantly, useful.

Providing a natural experience of telepresence is a multi-faceted problem. Not only do such systems have to provide natural means to perceive the surroundings, they also have to allow users to easily influence their surroundings. A remote system that allows for such capabilities can be thought of as a *'proxy'* to the user. Lack of any such capability can be equated to your proxy being blind, mute, deaf or amputated. Therefore, in order to simulate the experience of "being there", efforts have to be made to create such an immersive environment that fully stimulates all our senses. As we shall see, several academic and commercial initiatives have been taken to make telepresence more natural and immersive.

The quality requirements of such systems are strained by the fact that, the more engaging the system is made, the more susceptible the user is to inaccuracies in the system. To illustrate this, consider recording and playing a song in stereo instead of mono. This would not only require the use of two microphones and speakers, one needs to ensure that the two parallel recording and playback systems should have similar characteristics. Should this not be the case, and say, the sound to one ear is delayed with respect to the other; the user will experience discomfort. Thus, creating an immersive

system is not a case of simply extending existing technologies in a plug-and-play manner. As we shall see, in order to create such a system with existing technology, trade-offs have to be made to accommodate for the limitations of current technologies.

1.1 Context and Motivation

Binocular vision allows us to see in three dimensions. Each of our eyes sees the same scene from different angles. The same object is present in these two views at different positions. These two views are combined in our brain and the resulting process is called *stereopsis*. This along with several depth-cues we receive directly or indirectly from our sense-organs help us create a three dimensional picture of our surroundings, resulting in depth-perception.

As sight is our leading sense, it follows naturally that any synthetic environment would not appear natural as long as the system does not excite our vision to its fullest. However, common display systems strip us of the advantages of our ability to perceive depth via stereopsis. Displays that present a different view to each of our eyes, i.e. stereoscopic displays, can unlock this ability. There are many such displays in the market and many alternative technologies are being researched. Growing interest and recent developments in technology has made these displays cheaper than before and much more easily accessible. Like any other nascent technology, there is a wide array of technologies for achieving the same results. However, the sheer amount of data and processing requirements for capture, compression and transport of 3D video often exceeds the capabilities of economically viable embedded systems. Therefore, a simple, cheap, robust and easily reproducible toolchain for end-to-end stereoscopic remote vision can be beneficial for academic and industrial communities alike. This thesis is an effort to research ways to leverage established technologies that can be used to create end-to-end stereoscopic remote vision systems while satisfying the specific requirements of our desired application.

Consider a disabled person seated on a wheelchair (Figure 1.1). Currently, electric wheelchairs allow for easy mobility of aged and disabled people. However, performing daily tasks can still be a daunting task for people with limited upper body control. Further, constantly being seated also diminishes their reach. These people often find themselves in need for support. With the use of modern technologies, it should no longer be necessary for support personnel to always be present to aid them. Electric wheelchairs fitted with stereoscopic cameras and robotic arms can allow for remote assistance. Our target is to create this telepresence system to help improve the quality of life of our aging population.

1.2 Requirements

The several processes involved in such a toolchain may be coarsely grouped as Capture, Compression, Transport, Decompression and Display. Designing this an end-to-end so-

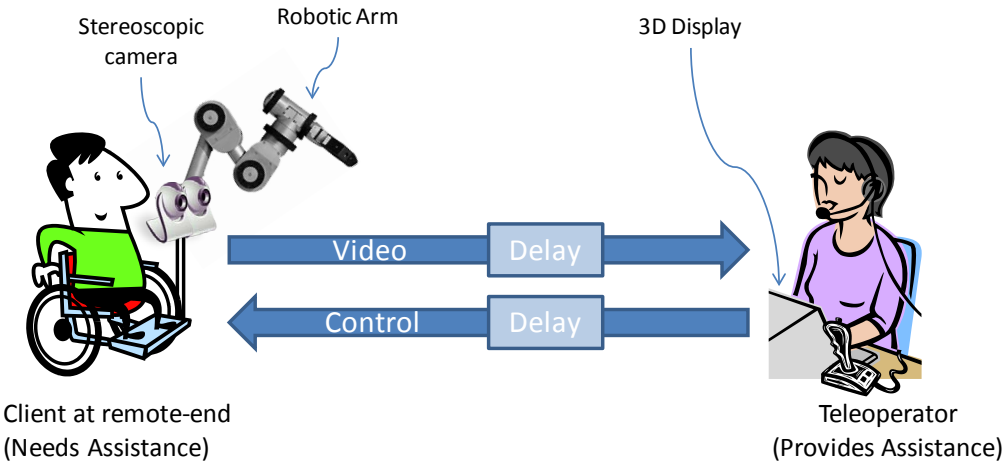


Figure 1.1: Schematic of assisted living, an application of this work. The camera and the robotic arm are together constitute the 'proxy'.

lution for stereoscopic video requires considering several alternative technologies present for a multitude of constituent steps in the remote vision toolchain. Each alternative method for carrying out these steps can have its advantages and disadvantages. Therefore, in order to select the best suited technologies for an intended system, it is important to gather a set of requirements for the system as a whole. Although many of these requirements can be resource or technology oriented; the most important requirements are derived from human-comfort and usability criteria. A brief set of these requirements is presented here:

1. **Immersion**: The more immersive a system, more easy it is for the viewer to believe that they are actually present at another place. Adding depth perception is one way to provide an immersive experience. This is the principle requirement of the system. Most of the other requirements are derived from this requirement.

2. **Latency**: The end-to-end delay between an event occurring before the proxy and the corresponding display of this event to the user should be minimum. This is essential for most applications of telepresence. For example, the difficulty of carrying out a conversation on the phone grows with the delay. This is even more important in the scenario where the user must manipulate the remote environment.

3. **Bandwidth**: Lower bandwidth consumption increases the possibility of ubiquitous use of a remote vision system. It also influences the cost of using such a system. Further, using lower bandwidth proportionally reduces network induced latency.

4. **Video Quality**: Higher quality video corresponds to better visibility and hence usability. Video quality is influenced by compression level, resolution and several other factors.

5. **Jitter**: Data traveling over networks can suffer from variations in latency (jitter). High jitter not only affects user comfort, it can drastically influence the usability of the system in certain applications.

6. **Left-Right View Correlation**: As we shall see, even slight deviation in the left and right views can have drastic influence on the human comfort levels. This places stringent requirements on the relative position of the cameras.

7. **Size**: The proxy has to be an embedded system that can easily fit on a wheelchair. Small size also means that there is limited processing power available at the proxy.

8. **Cost**: As with any other system, cost is a major factor for the suitability of the remote vision system for its various applications. Most importantly, remote assistance can be provided to multiple individuals from a single teleoperation machine. This many-to-one relationship means that the 'proxy' system should be low cost while a costlier teleoperation system can be used.

In addition to the above requirements, we must also minimize development time while ensuring reliability.

1.3 Structure of this Report

The rest of this report is organized as follows: Chapter 2 provides some background information about the different aspects of stereoscopic vision and related areas of research such as teleoperation and human depth perception. It also provides information from which more specific requirements can be derived. Chapter 3 discusses design choices made by comparing the pros and cons of the several alternative technologies involved at every step of the targeted toolchain. It also covers the architecture and implementation specific details of the system. Chapter 4 provides a case study of the current solution. It describes characteristics of the system discovered by carrying out meticulous measurements. Further, it also discusses a user survey conducted to discover impact of variations in the system's characteristics upon human operator performance in a teleoperation like scenario. Finally, Chapter 5 concludes this report and talks about parts of the system where there is a room for improvement or extension.

2 Background

"Knowledge is indivisible. When people grow wise in one direction, they are sure to make it easier for themselves to grow wise in other directions as well. On the other hand, when they split up knowledge, concentrate on their own field, and scorn and ignore other fields, they grow less wise even in their own field." – Isaac Asimov

A large part of this thesis has been an effort to bring together the knowledge of several fields of science. This chapter attempts to collate the information gathered to better understand the requirements of a stereoscopic remote vision system intended for telepresence applications and to explore and evaluate the several alternatives to develop such a system.

2.1 Telepresence and Teleoperation

Telepresence is a term used for technologies that allow people to assert their presence at a remote location. As mentioned before, the degree of presence is enhanced if more of our senses are stimulated by the system. Correspondingly, it can be expected that the usability and effectiveness of the system would increase as well. This idea has inspired several initiatives to create telepresence systems that allow humans to perceive depth at the remote location. A large portion of research being carried out in this field aims to take the current teleconferencing systems to the next level by allowing for depth perception. Some examples are the NTII[42] and VIRTUE[26] projects.

Teleoperation is the handling of machines from a distance. Remote controlled robots have been in use in space and military for a long time. More recent applications include the Da Vinci robot that allows surgeons to perform minimally invasive surgeries with a very high degree of precision with robotic actuators while looking through a stereoscopic display. Stereoscopic remote vision systems have been shown to significantly improve human performance in teleoperation [16, 53, 35, 15]. Figure 2.1 shows the results of one such study comparing the performance of human operators with stereoscopic and monoscopic video for a peg-in-hole task. The objective of this thesis is to develop a stereoscopic remote vision system intended for telepresence applications. The envisioned application is to provide "assisted living" services to disabled and aged people with the help of the stereoscopic remote vision system coupled with remotely controlled actuators.

2.1.1 Delay in Teleoperation

Delay is known to significantly degrade human performance in teleoperation tasks. Research conducted at NASA suggests that: "Delays as small as 1/4 second are

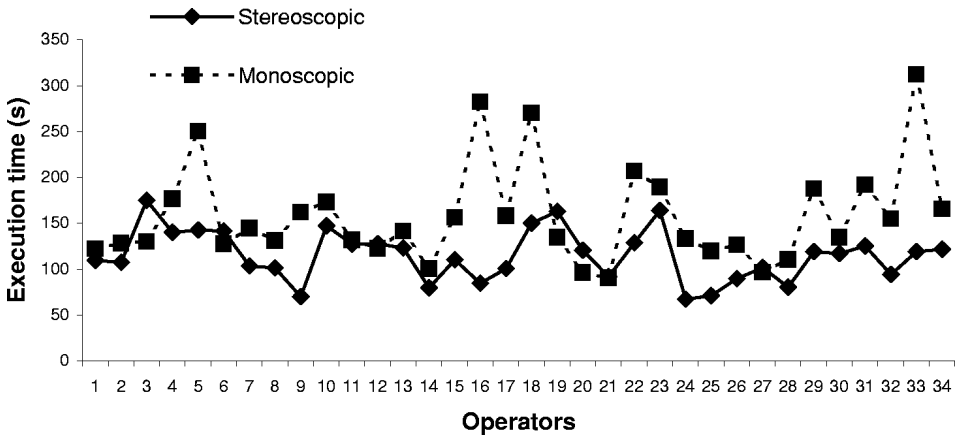


Figure 2.1: Comparison of Stereoscopic and Monoscopic video for teleoperation [16]

noticeable to the operator. Delays as small as 1 second, measurably degrade the operator's performance."[7]. Multiplayer computer gaming is another scenario similar to teleoperation where delay plays a significant role in the quality of gameplay. In this context, Smed et. al. write: "Latency affects the users performance nonlinearly: continuous and fluid control is possible when the latency does not exceed 200 ms, after which the interaction becomes more observational and cognizant."[58]. Consequently, significant effort has gone into mitigating the effects of delay.

A majority of teleoperation systems use special control schemes to compensate for operator inaccuracy that results from delay. An example of such schemes is compliance control, where the stiffness of the robot's actuators is reduced when it hits an object. A comparative study of these control schemes is presented in [3]. Another method is to use predictive display where 3D models of the actual robot are superimposed on the incoming video [6]. Thus, the operator receives immediate feedback about the model's action corresponding to his/her control followed by that of the actual robot. Although these techniques provide significant improvement in operator performance, it remains important to minimize the actual delay wherever possible. As we shall see, minimization of delay has been an important factor in the selection of technologies and their implementation throughout this project.

2.2 Monoscopic Remote Vision

Before we discuss stereoscopic / 3D vision and related research and technologies, it is required that we lay some groundwork about monoscopic remote vision. Here, we describe the several transformations a scene undergoes before it is shown to the viewer. We shall describe the major steps of Capture, Compression/Decompression, Transport and Display and the processes involved in these steps that are relevant to our specific problem and its solution.

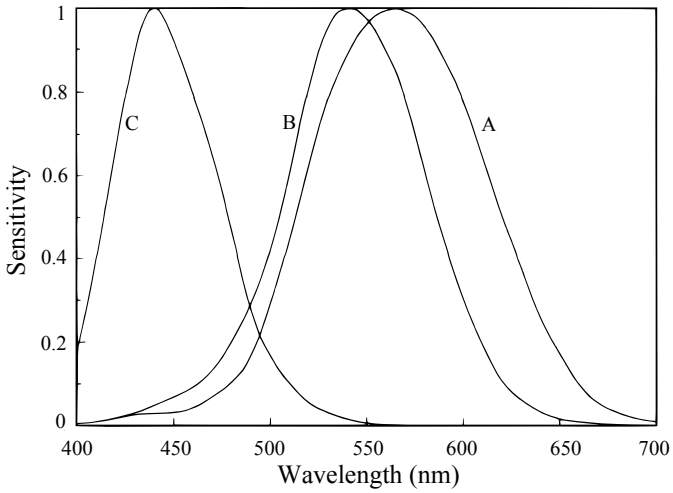


Figure 2.2: Sensitivity of the three types of colour receptors in the human eye[47] [1]

2.2.1 Capture

Digital images are essentially two dimensional grids consisting of points that represent colours. Digital cameras use arrays of sensors to translate an actual scene into a digital image. Each of these sensors corresponds to a pixel in the resulting image. Typically, these sensors are sensitive to a large spectrum of light, therefore, in order to obtain a color image, colour filters are placed before these sensors. Digital cameras can use a variety of pattens for the color filters, one of the most common patterns is the Bayer Color Filter Array [45] as shown in Figure 2.3. Since the sensitivity of the human eye varies with the frequency (i.e. colour) of light [40], the Bayer pattern contains twice as many pixels for green as red or blue. A camera sensor chip reads the intensity of light at each of these sensors and converts them into digital values.

At this point, each pixel only contains intensity information corresponding to the colour of the filter at that location. Therefore, the other two colours must be estimated at each pixel to make the information complete. This process is called demosaicing [19]. A relatively simple method to perform demosaicing would be to use the average values of nearby pixels. This demosaicing can be done by means of a simple bilinear interpolation.

For example, the Red, Blue and Green Values at Pixels B2 and G5 can be calculated as shown in Equations 2.1 through 2.4. A Green value at a Red/Blue pixel is interpolated as the average of the four nearby green pixels. A Red/Blue value at a Blue/Red location is the average of the four diagonal Red/Blue pixels. Finally, a Red/Blue value at a Green pixel is the average of the two nearest red/Blue pixels.

[1] The curves in Figure 2.2 are normalized to unity. The peak sensitivity of the green-absorbing receptors (B cones) is approximately 5% higher than the red-absorbing receptors (A cones) and 30 times greater than the blue-absorbing receptors (C cones) [47]. Please note that this figure is an enhanced replica.

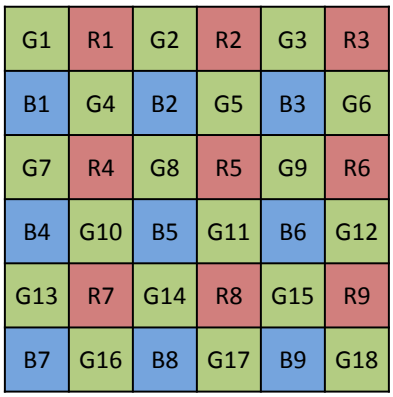


Figure 2.3: The Bayer color filter array

$$Green(B2) = \frac{G2 + G4 + G5 + G8}{4} \tag{2.1}$$

$$Red(B2) = \frac{R1 + R2 + R4 + R5}{4} \tag{2.2}$$

$$Blue(G5) = \frac{B2 + B3}{2} \tag{2.3}$$

$$Red(G5) = \frac{R2 + R5}{2} \tag{2.4}$$

Demosaicing can lead to false colours and artifacts in the image. Several advanced demosaicing algorithms are discussed by Gunturk et. al. in [19]. However, most of the commercially used demosaicing algorithms are closed proprietary information. Battiatio et. al. [5] have discussed some recent patents in this area.

2.2.2 Compression and Decompression

Compression is a widely used method for reducing the size of audio-visual data, allowing for reduced storage space and transmission time. The scope of this thesis is limited to video compression and decompression. Since video codecs are extensions of still image codecs, we will discuss still image compression first.

2.2.2.1 Still Image Compression: JPEG

A simple image compression algorithm can exploit the fact that large sections of images are spatially correlated i.e. nearby pixels in a digital image typically contain the same data. Therefore, it is possible to compress an image with a generic entropy encoding algorithm. However, this compression can be improved by exploiting the limitations of human vision. Figure 2.4 shows the response of the human vision system to variation in light intensity with respect to the angular frequency (measured relative to the observer's eye) of these changes. This is indicative of two facts:

1. Humans are less sensitive to finer variations in light intensity.

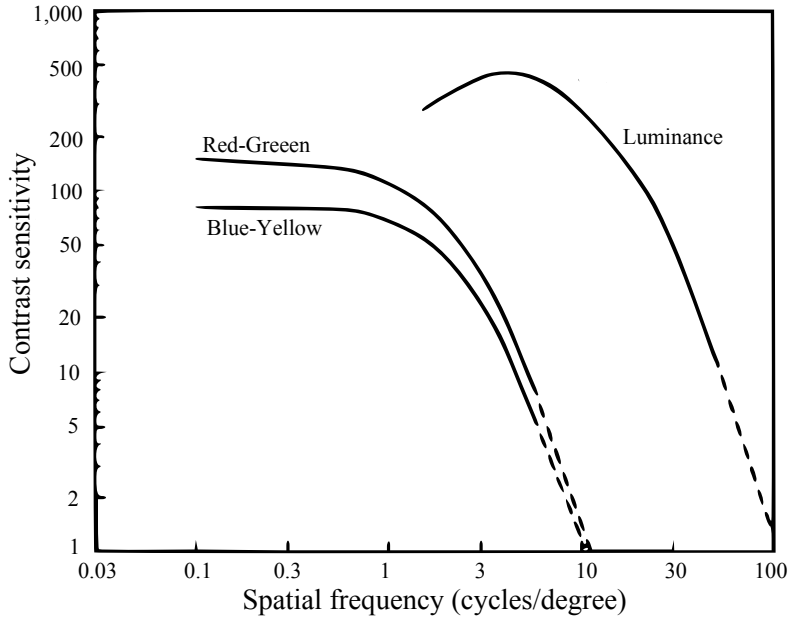


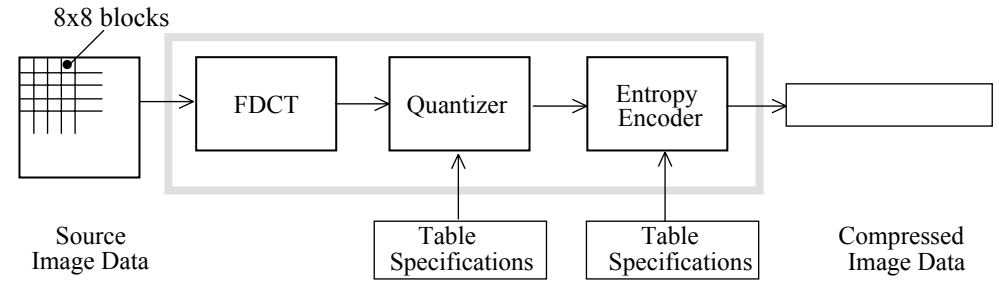
Figure 2.4: Human vision sensitivity to changes in luminance and chrominance intensities[2] [47, 48]

2. Humans are less sensitive to colour variation as compared to intensity variation.

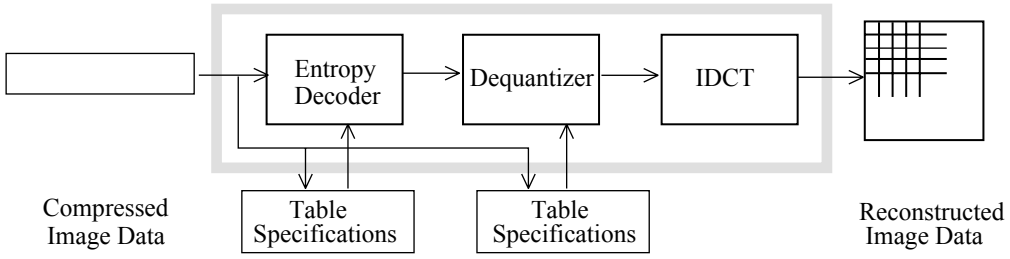
The JPEG Still Picture Standard is once such standard for systems that exploit these characteristics. It has become a de-facto universal method for compressing/decompressing still images for storage and transmission. The JPEG compression takes advantage of these characteristics by preferentially compromising the quality of details in an image while retaining its overall structure. The major steps involved JPEG Compression and Decompression are depicted in Figure 2.5. After a possible colour space transformation and downsampling, the incoming image is divided into blocks of 8x8 pixels. This block is transformed into its frequency-domain representation with a Discrete Cosine Transform. Once the image is in the frequency domain, it is possible to individually process its different levels of spatial detail. This means that we can decide to remove the finer details of the image while retaining its overall structure. This is done with the help of the quantization process where different frequencies can be divided by different factors (stored in a quantization table). This data is then encoded with an entropy encoding (typically Huffman) algorithm.

The level of desired quality (and therefore the output size) can be altered for example by using a different quantization table. A quantization table containing larger divisors would result in less variation in the input data to the entropy encoder, thereby reducing the output size. A lower quality requirement also means that faster, less accurate algorithms can be used to perform the several steps involved in the compression process. Further, a smaller image takes less time to transfer over a network. Therefore, a trade-off

[2]Figure 2.4 is an enhanced replica.



(a) The JPEG Encoding Process



(b) The JPEG Decoding Process

Figure 2.5: The JPEG compression / decompression processes [62]

exists between size, speed and quality of the image. As we shall see, this trade-off also exists for video encoding, decoding and transfer.

2.2.2.2 Video Compression

A simple video compression algorithm would simply compress each individual frame of a video with the JPEG algorithm. However not only are the frames in a video spatially correlated within themselves, they also have a temporal correlation between the frames i.e. successive frames in a video are usually small transformations of the image in the past and future frames. Video compression methods exploit this fact to obtain better compression ratios.

Although there is a large variety of video codecs that can be used to compress a video, they all make use of nearly the same principles to achieve compression. For this discussion we use the MPEG codec family as a reference. The MPEG codecs exploit the temoral correlation between successive frames by estimating motion of objects in nearby frames with respect to each other. This process is called motion compensation[47].

The MPEG codecs divide the frames into blocks whose motion is estimated. These blocks are called *macroblocks*. A search algorithm is used to match a macroblock in the current frame to another macroblock in a reference frame. Next, a motion vector is calculated for this macroblock. This vector is essentially the x and y translations of the macroblock with respect to its reference. For example, the object in Figure 2.6 has been

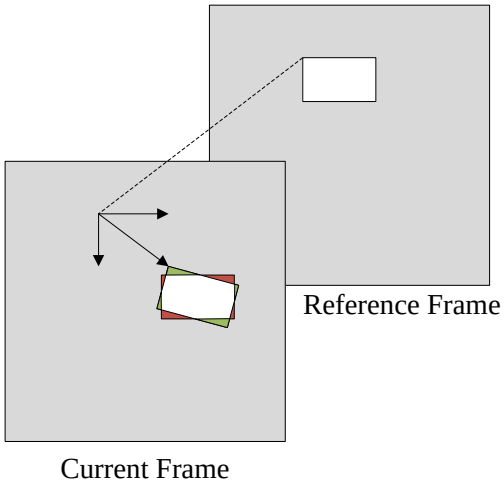


Figure 2.6: Motion Compensation

moved and rotated in the current frame. Since this is the only change in the image, the rest of the picture need not be stored/transmitted again. The codec only sends the motion vectors and if required, the residual error image (coloured portion in the figure) for the current frame.

If a certain macroblock is not found to have changed in between frames, then no new information needs to be sent/encoded for this block. However, it is possible that the video transforms such that no match between a macroblock and the available reference macroblcoks is found. In this case, this macroblock is compressed independently.

As shown in Figure 2.7, three kinds of frames are used in the MPEG codecs:

- **I Frames** or *intra* coded frames are the frames that are processed independently of other frames in the video stream. The compression used is principally similar to the JPEG compression. Therefore, these frames do not utilize the temporal correlation themselves. However, these frames are used as reference frames to compress the other two types of frames.
- **P Frames** or *predictive* coded frames utilize the temporal correlation using predictions obtained while keeping the nearest preceding P or I frame as a reference.
- **B Frames** or *bidirectionally* predictive are essentially predictive frames that use the preceding and/or upcoming I or P frames as reference frames. However, in H.264, B frames can also be used as a reference for other frames.

Typically, the three kind of frames are sent out in a repetitive pattern called a Group of Pictures (GOP). For example the GOP in Figure 2.7 is IBBPBBPBB. However, a certain GOP structure is not always followed because of scene changes in the video to be encoded. If a scene change occurs at a certain frame, the content of that frame is usually completely different from the past frames. Such a frame must be encoded as an I frame.

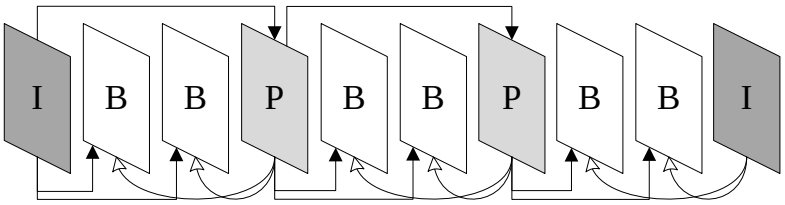


Figure 2.7: Prediction in MPEG Frames

The MPEG codecs do not specify any specific searching algorithm for matching the macroblocks. However, typically the algorithms used for this purpose are fairly complex and computationally expensive. Further, any system using P or B frames should be capable of buffering the required number of frames to be able to exploit the temporal correlation between frames. Further, in order to display a B frame, the future I or P frame should already be available to the display system. This will cause a delay in the case of live streaming of video. Because of such constraints, it is not always possible to make use of temporal correlation. In such cases, each frame may be compressed individually. MotionJPEG is one such class of video codecs that compress each video frame individually.

2.2.3 Data and Video Transport

Layer	**Protocols**
Application	HTTP, DNS
Presentation	SSL, TLS
Session	PPTP, RPC
Transport	TCP, UDP
Network	IP, IPsec
Data Link	PPP, Ethernet
Physical	Ethernet, RS-232

Table 2.1: The OSI Reference Model

The structure of commonly used protocols over the internet loosely follows the OSI Reference Model (Table 2.1). In order to keep the cost of networks low, the more numerous intermediate devices that are responsible for connecting machines, have to be cheap and therefore dumb. Consequently, the lower layer protocols have to be simple and complexity must bubble upwards into the end-points: computers.

As shown in Figure 2.8, the internet protocols follow an hourglass like structure. The Internet Protocol (IP) layer binds together the multitude of layers above and below it, allowing them to inter-operate. It is these characteristics of this design that allowed the internet to spread and become as ubiquitous as computers and portable communication

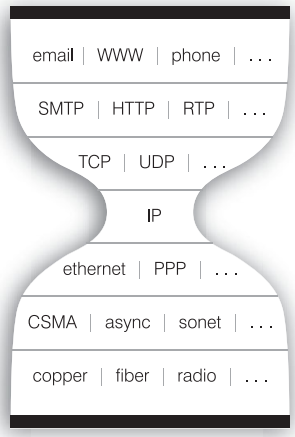


Figure 2.8: Hourglass structure of the Internet [67]

devices. However, these very qualities also cause computer networks to be unreliable and non-deterministic. For example, because of the hourglass structure, there is no way for an application to know what physical medium is used to transport any data it sends or receives. This means that the application doesn't have any knowledge of network behavior that may affect its performance. It can only hope to characterize the network by making some measurements. Further, because the complexity of interconnecting devices has to be kept low, it is impossible to guarantee

- Hop-to-hop data integrity
- Preservation of packet order
- Data rate
- Latency and Latency Jitter

at the IP Layer. Accounting for these conditions and correcting where possible and necessary is left up to the higher level layers.

Such characteristics cause computer networks to be an imperfect match for real-time applications such as remote vision. Given these conditions, it is important to decide the level at which these deficiencies should be compensated for. The choice of using TCP or UDP protocols becomes important at this point. These protocols offer either reliability and congestion control or timeliness. As we shall see, for real-time multimedia applications, timeliness is more important. The lack of congestion control in UDP can cause congestion breakdown in case of heavy traffic. Datagram Congestion Control Protocol (DCCP) is a next-generation transport protocol that allows for congestion control like TCP but does not ensure reliable in-order delivery. However, since DCCP is a proposed protocol, not many successful implementations of DCCP are available.

Although it is not uncommon for multimedia applications to transport audio and video streams directly using the TCP or UDP protocols, several important services are required by real-time media applications that these protocols fail to provide. Therefore,

it is usually beneficial to use a protocol designed for real-time data. One key standard protocol for audio/video transport over IP networks is the Real-Time Transport Protocol[55] (RTP). Some of the several important services offered by RTP are timing recovery, loss detection/correction and media synchronization [49]. Further, since timeliness is more important for streaming media, RTP is usually coupled with UDP. As we shall see, because of its several advantages, RTP was also chosen for our specific implementation for transporting live stereoscopic video.

2.2.4 Display

The present day consumer electronics market boasts products based on a wide variety display technologies and many others are under development. However, a discussion on the several display technologies is beyond the scope of this thesis, instead we discuss some of the common issues with displays that affect the user's comfort.

CRT based displays update the contents of the screen periodically by means of an electron beam progressively moving across the horizontal lines on the screen. The number of times this electron beam completes this process for the whole screen in a second is called the *refresh rate*. Although CRTs are rarely used in the present day, and the more common LCD display's do not progressively scan the screen; the notion of refresh rate is still used. It can now be equated to the rate at which the video card sends out the contents to be displayed at the screen.

Typically, a computer display system can be thought to be asynchronous. Applications place the content to be displayed on the screen on a memory location. This can be thought of an application *drawing* on the screen. Meanwhile, the display is updated periodically. Because the process of drawing on the screen occurs independently from the display process; it is possible that the user can see the drawing process in progress, called the tearing effect.

In order to avoid the tearing effect, a double buffer is used. The applications draw on a 'back buffer' while the contents of the 'front buffer' are shown on the screen. When the drawing process is complete, the contents of the back buffer are copied to the front buffer during the blanking period of the screen (time during which no new data is sent to the screen). Alternatively, the roles of the memory locations assigned to the buffers may be swapped. This is called *page flipping*. In general, to avoid the tearing effect, the read and write operations of a frame must be exclusive at several steps in the complete end-to-end system. As we shall see, this adds to the delay suffered by the video in our remote vision system.

2.3 Depth Perception

Our eyes are positioned such that our brains receive similar images of a scene taken from two nearby points, usually positioned at the same height. If two objects are present at different depths from the viewer, the relative positions of their images in the two

eyes will differ. Our brains measure this disparity and use it to estimate depth [50, 36]. This process is called *stereopsis* and the difference in the two images is called *binocular disparity*. Stereopsis is commonly confused with *depth perception*. This is inaccurate as our brains rely on several cues for depth perception[32, 30, 20, 27]. The major visual cues are:

- **Binocular Cues**
  - **Stereopsis**: Disparity in two parallel, horizontally separated views of the same scene.
  - **Convergence**: In order to focus on an object, our eyes converge inwards, bringing the object at zero disparity. The degree of this convergence gives a sense of an objects proximity to the observer. The terms *vergence* and convergence are used alternatively in literature and this report.
- **Monocular Cues**
  - **Kinetic Depth Perception**: The perception of distance from objects by change in their size as they move towards or away from the observer.
  - **Motion Parallax**: The apparent relative motion of objects seen by an observer as they move.
  - **Retinal Image Size**: Objects of the same size appear bigger when they are closer. This is even more helpful when the objects are familiar to the observer.
  - **Occlusion**: An object in front of another blocks the view to the one behind.
  - **Aerial Perspective**: Scattering of light by the atmosphere causes objects at a distance to have less contrast, luminance and colour.
  - **Accommodation**: The sensation of contraction and relaxation of ciliary muscles as we focus on objects at different depths. This is only effective for objects close to the oberserver.
  - **Texture Gradient**: The decrease in clarity of an object's texture with distance.
  - **Light and Shadows**: The way light falls on an object, its reflection and shadows.

As shown in Figure 2.9, the several depth cues are active at different ranges of distance. It can be discomforting if our brain receives conflicting information from independent depth cues. Therefore, an artifical system that exploits one or more of these cues should take into account the strength of these cues at its distance of operation. This knowledge can also be used to dynamically adjust the system according to the content of the scene.

Other than our eyes, the vestibular system (the balance organ inside our ears) also contributes to our depth perception [31, 51, 18, 43]. With motion parallax we can only estimate the relative depth of objects in our vicinity. When our brains combine this

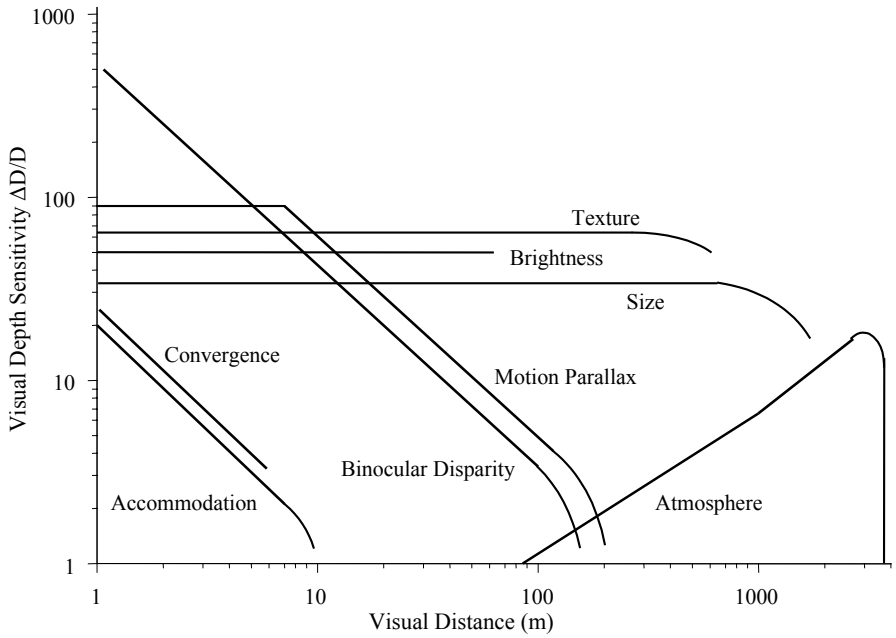


Figure 2.9: Effectiveness of Depth Cues vs. Distance. Derived from [44]

with the the movement and orientation information of our own bodies (obtained by the vestibular system), it can create an absolute map of our surroundings. This explains how people with only one functional eye, and animals lacking binocular vision can perceive depth.

2.4 Stereoscopy: Evolution of technologies

Stereoscopy is the process of creating the illusion of depth in an artificially created scene. It exploits the process of stereopsis. The viewer(s) is presented with two or more similar images of the same scene while ensuring that each of the images is only visible to the eye it is meant for.

Figure 2.10 shows a timeline of major events pertaining to stereoscopy. The first stereoscope was invented by Sir Charles Wheatstone in 1838. As shown in Figure 2.11a, the device employed an arrangement of mirrors to present two similar drawings that created the illusion of depth. With the advent of silver plate photography, these drawings were replaced with photographs and stereoscopes became popular.

Later, anaglyphs were used for black and white and colour photographs. For viewing an anaglyph, the audience wears red/green or red/cyan filters over their eyes and the photograph contains depth in the form of disparity in the same colours. This technology took stereoscopy to the world of cinema, generating interest and investment in relevant technologies. However, the separation of images meant for the left and right eyes was not very good and the audience usually left the halls feeling nauseated. This lead to

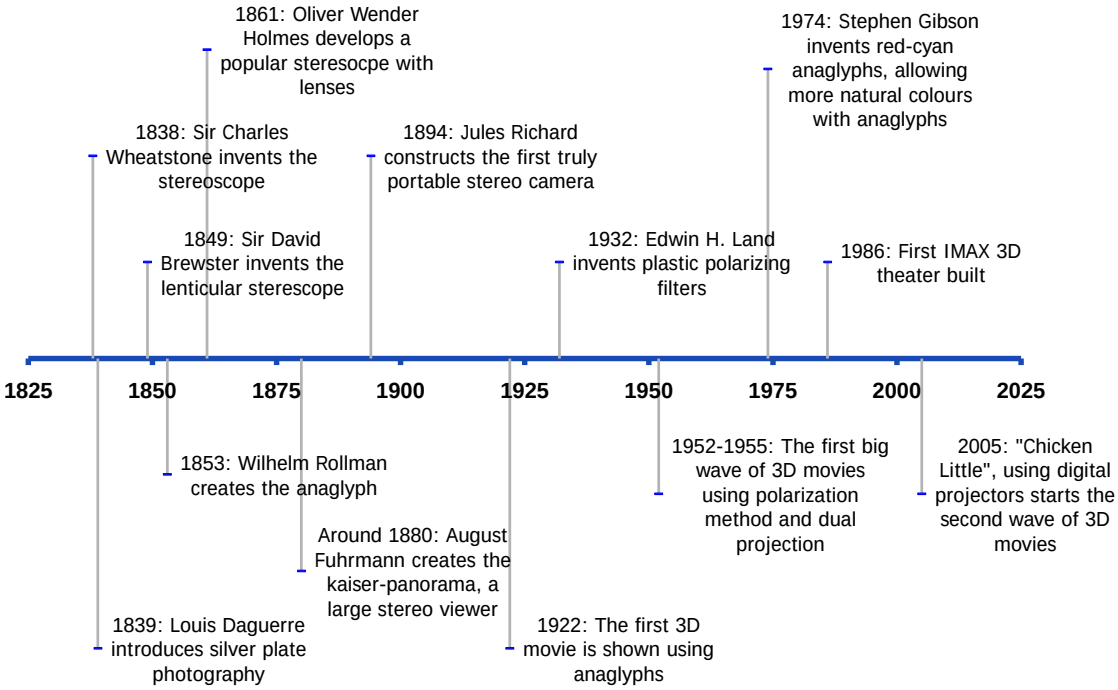


Figure 2.10: A History of 3D Media

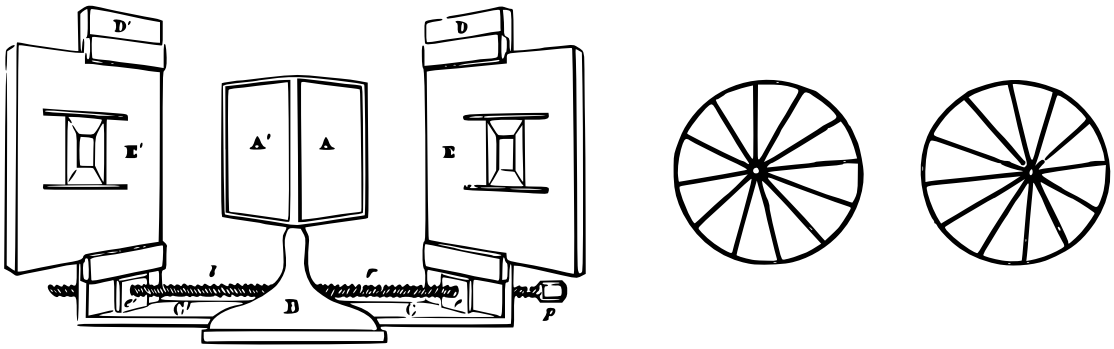
invention of new technologies to create the illusion of depth. These current technologies for stereoscopy are discussed in later sections.

2.5 Stereoscopic/3D Remote Vision: Contemporary Technologies

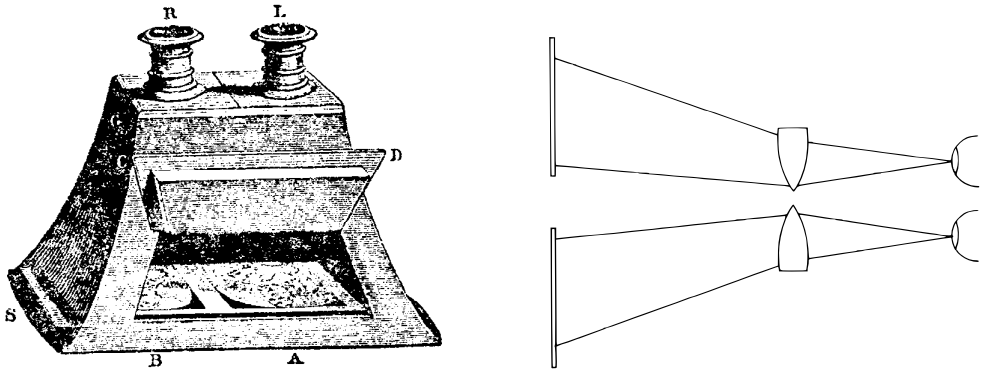
Here we discuss the several technologies involved in stereoscopic/3D remote vision. Being an end-to-end solution and an area of active research, it has a large multitude of technologies involved. Therefore, a lot of options need to be explored when designing such a system. Here, we briefly describe the principles behind each method while briefly discussing their pros and cons. A comparative study of these alternatives is provided in a later chapter.

Much of the research in this area is being driven forward by the 3DTV market. Consequently, much of the solutions are geared towards fulfilling the requirements of 3DTV. This makes choosing an appropriate set of technologies more difficult for our particular application. This is further complicated by the interdependence amongst the technologies involved at the different steps of the solution as one technology might be an ideal match for one step and its counterpart unsuitable for another. A discussion of transport technologies is omitted here as the technologies used in the context of the

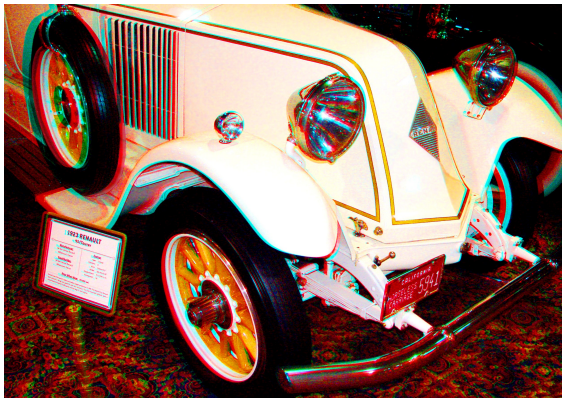
[3]Some figures are enhanced replica.



(a) The first Stereoscope and a drawing used for it[32]



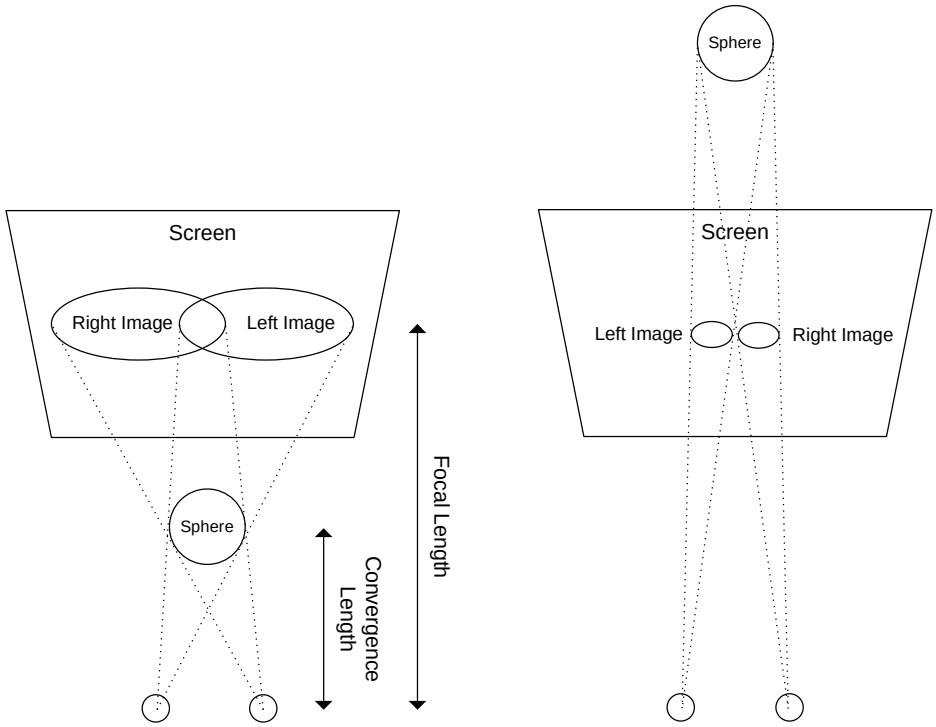
(b) The Brewster stereoscope and its optics.[32]



(c) An anaglyph photograph of a 1932 Renault

Figure 2.11: The Evolution of Stereoscopy [3]

internet do not differ from those for monoscopic video. Moreover, a discussion on display technologies is provided first to allow for the establishment of basic facts.



(a) Object seen in front of the screen (b) Object seen behind the screen

Figure 2.12: Binocular Disparity in stereoscopic displays and accomodation - convergence conflict

2.5.1 Display

Television is one of the largest consumer electronics markets worldwide and the prize for developing a successful display technology for 3D video is likely to be great. Consequently, there has been a large push from the industrial and academic communities to come up with a multitude of candidates for the next generation display technology. The success of any technology in this realm depends on its technical viability, cost and user comfort. So far there is no clear winner in this field. Apart from the criteria mentioned before, the choice of a display also rests on its compatibility with the technologies in the complete end-to-end system, development effort and scalability. The discussion offered here attempts to cover a majority of the contemporary 3D display technologies. More comprehensive discussions on these technologies are present in [8, 37, 23, 46].

Most displays render two or more images and use some mechanism to ensure that the user's eyes only see the view meant for them. Figure 2.12 shows this process. A sphere in the original scene is displayed on the screen. If the right image of the sphere is displayed to the left and the left image to the right, the sphere is perceived in front of the screen and vice versa. The user's eyes converge to see the object while they are always focused on the surface of the screen. This leads to a mismatch between the depth cues

perceived by our brain, leading to user discomfort. This is a common shortcoming of most displays discussed here. The only exceptions being Volumetric and Holographic displays. The problem has proved to be very difficult to crack and is generally circumvented and not solved. A detailed discussion of human comfort is presented later. Broadly, these display systems are categorized as binocular (showing only two views) or multi-view and autostereoscopic or those requiring user-mounted objects such as glasses or head-tracking devices.

2.5.1.1 Field Sequential Technologies

Field sequential or time multiplexed systems typically alternatively show left and right views and use a blocking mechanism to ensure that the left eye cannot see the image meant for the right eye and vice versa. Since these systems only show two views, they fall under the category of binocular systems. Historically, mechanical devices were used for blocking but current systems generally employ LCDs. The user wears goggles that have an LCD in front of each eye. The screen, refreshing at a high rate, alternatively shows the left and right views while the LCD shutters are opened and closed to allow only one eye to see the screen at a time. Typically infrared emitters attached to the display system send pulses to synchronize the glasses with the display. CRT screens were typically used as displays but recent advancements in LCD technologies have pushed their refresh rates high enough to be used for this purpose.

An example of such systems is nVidia's 3D vision system. The company provides support for connecting a pair of LCD shutter glasses and an infrared emitter to their graphics cards via USB or via the VESA stereo interface. Several screens supporting high refresh rates can then be coupled with the system. The system is capable of rendering two separate views for 3D games. Further, it allows use of the standard OpenGL library to manipulate separate buffers for the left and right views. Other examples include DLP®-3D[25] based HDTVs offered by Samsung and Mitsubishi.

Modern LCD screens can operate at 120Hz so each eye receives images at 60Hz. However, shutter glasses can have synchronization issues at high refresh rates. Further, the shuttering of glasses can develop beats with artificial lighting. An advantage of this technology is that the system can easily switch between normal and stereoscopic operation. However, these systems can suffer from ghosting effects especially for bright areas in the scene as the switching time of an LCD pixel increases with brightness.

2.5.1.2 Polarization based Technologies

Another method in the category of binocular systems is to use the principle of polarization of light to separate the left and right views. In such systems, the source produces left and right views of orthogonally polarized light while the user is required to wear polarization filters to isolate the left-right views. The principle is applied in a variety of methods. The simplest of these methods employs two projectors, one stacked on top of the other. Light coming from the projectors is linearly polarized with the help of

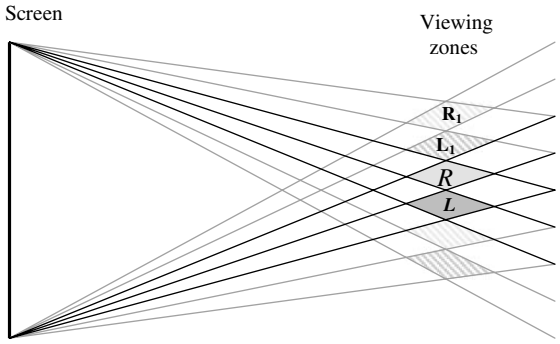


Figure 2.13: Viewing zones for autostereoscopic displays [46]

filters[4]. Because of the difference in location of the two projectors, dual projector based systems suffer from the keystone effect.

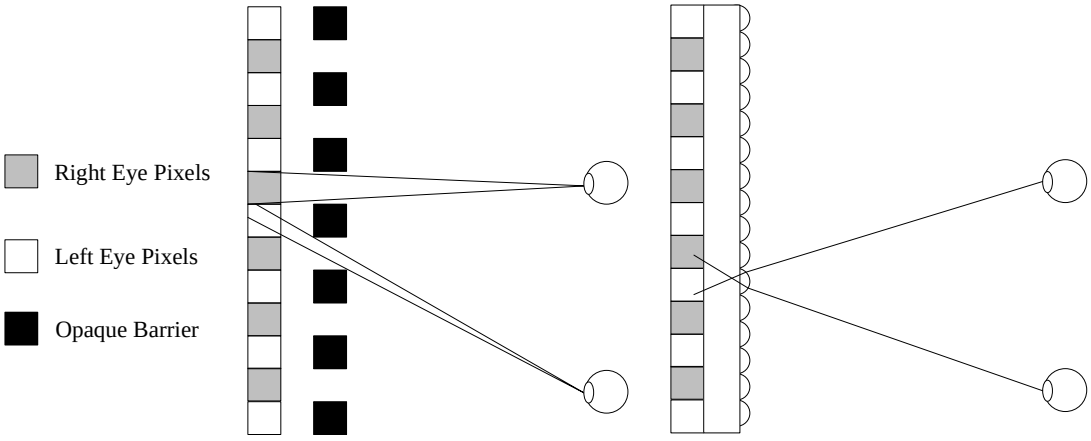
Other systems such as the LCD monitors produced by Zalman use circularly polarized light instead of linear polarization. These monitors circularly polarized alternate lines in orthogonal directions. Clearly, interlacing reduces the vertical resolution for each eye by half. Another approach is to individually rotate the polarization of each pixel. Monitors manufactured by iZ3D work by rotating the polarization plane of each pixel individually. For example, if a pixel is to be seen as white by the right eye and black by the left, the linearly polarized light of that pixel is rotated to be parallel to the axis of right eye filter. Consequently, varying degrees of rotation can individually vary the brightness of each pixel. One disadvantage of the iZ3D system is ghosting of images caused by imperfect polarization.

Most polarization based systems use linearly polarized light. A clear disadvantage of these systems is that the brightness perceived by the user depends on the angle of user's head with respect to the screen. Imperfect polarization also results in ghosting of images. Further, projector based systems also require a special screen. Even though the initial set up costs of such systems is high; the filter glasses are very cheap. Therefore, the cost of adding viewers is very small. Consequently, these systems are widely used in cinemas.

2.5.1.3 Autostereoscopic Technologies

A major drawback of the previously mentioned techniques is that they require the user to wear glasses. Autostereoscopic displays create viewing zones in front screen where either the left or the right image is visible [46]. Figure 2.13 depicts these viewing zones, the user's eyes must be located in one of these zones to perceive depth. These displays produce the left and right images on alternating pixel columns. Next, separation of the left-right views is achieved by use of a parallax barrier, lenticular screen, controlled

[4]Light from DLP projectors is unpolarized. Therefore polarizing filters need to be placed in front of the projector. On the other hand, more efficient polarizers are required to be used in case of LCD projection.



(a) Parallax Barrier Screen (b) Lenticular Screen

Figure 2.14: Optics of Lenticular Screen and Parallax Barrier Methods

light sources behind the pixels and prismatic screens [46]. Size of a viewing zone is less than that of the separation between human eyes: around 65 millimeters. Figure 2.14 demonstrates the lenticular screen based methods. A lenticular screen consists of semi-cylindrical lenses placed vertically in front of the screen. Light coming from each pixel is refracted from these lenses, resulting in only one of the views being visible in a certain zone. Screens based on the parallax barrier method use fine vertical slits in an opaque screen placed in front of the pixels. Similar to the lenticular screens, only pixels corresponding to one view are visible from a given viewing zone.

A clear disadvantage of binocular autostereoscopic displays is the existence of viewing zones. Consequently, the user observes crosstalk[5] between views [28]. Say the user's left eye is positioned at zone R and the right eye is positioned at zone L_1. In this case, the left and right eyes see the views meant for each other. This can lead to a complete breakdown of the 3D effect. One solution to this problem is to use head tracking systems. In these systems, the pixel columns corresponding to the left and right views can be adjusted according to the position of the user's head. Further, viewing the system from different lateral positions gives a false effect of rotation of the objects and distortion of depicted objects. An advantage of binocular displays is that they demand a limited bandwidth and hardware from the transport and capture systems. Head tracking can also be used to display one of several views (to one viewer), extending binocular systems to multiview systems; thereby allowing the perception of motion-parallax. Lenticular screen and parallax barrier methods have been extended into multiview systems in in Philips 9-view and the Sanyo 4-view displays respectively.

[5] Crosstalk occurs when a percentage of the view meant for the right eye is seen by the left eye and vice versa.

2.5.1.4 Head Mounted Displays

Head Mounted Displays (HMDs) operate on the same principles as the Brewster stereoscope. The user wears the HMD device like goggles. LCD's placed in front of each eye display separate images to the left and the right eye. Since the LCD screen is very close to the eyes, the accommodation-convergence mismatch can be very pronounced. Lenses are used to increase the focal distance to values close to 50 centimeters [63]. HMDs have two differentiating factors: the user is completely isolated from his/her surroundings; the focal distance is fixed. Further, as most users are used to working with screens, they feel out of place whilst using HMDs. A survey of comparisons between screen based displays and HMDs is presented in [57] and the general conclusion is that HMDs generally induce more user discomfort than screens. Consequently, use of HMDs is limited to situations where portability is required.

2.5.1.5 Other Display Technologies

Other classes of display methods include volumetric and holographic displays. Volumetric displays actually reproduce the scene on a volume of space. Therefore, there is no conflict between accommodation and vergence. These systems employ optical and mechanical methods to reproduce *slices* of the actual scene [46]. These slices sequentially scan the volume of the scene. A disadvantage of these systems is their large size and the need for moving parts. As discussed before, holography captures the intensity and phase of the incident light in the form of an interference pattern. The Spatial Imaging Group at MIT were the first to attempt production of moving holographic pictures [46] by using acousto-optic modulators. Holography been an area of research for a long time but its requirements such as very high computation requirements, high amount of data involved, inability to capture ambient light make it impractical. However, distant as it may be , holography offers the promise of a future with true reproduction of a scene.

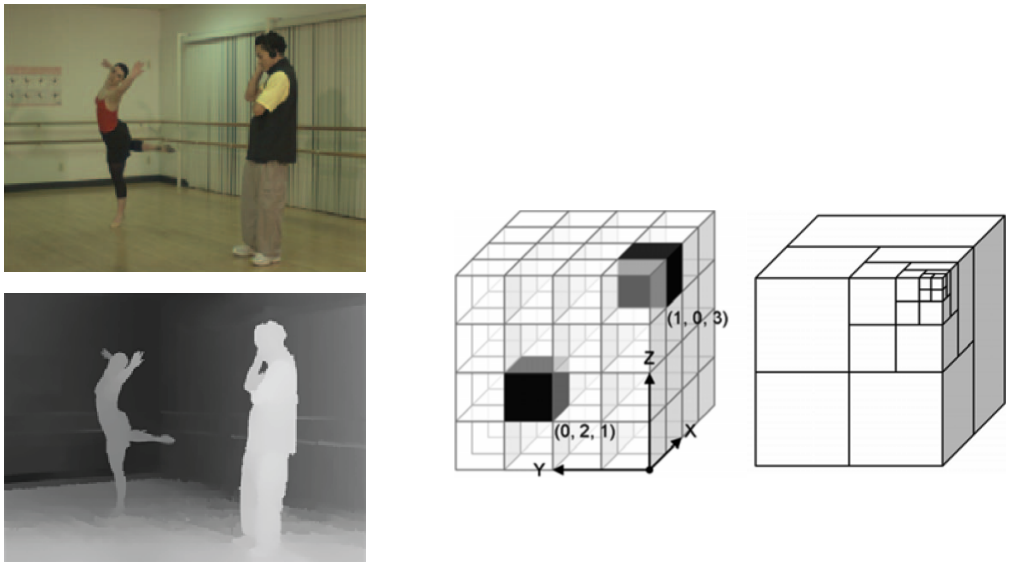
2.5.2 Capture

A large variety of methods have been developed to capture 3D or stereoscopic video and their characteristics vary over a large spectrum. A short discussion of these techniques is presented here.

2.5.2.1 3D Scene Representation

Because of the tight coupling of capture and representation methods, we touch upon some methods used to represent 3D scenes first. 3D video can be represented either in its raw original form, which in most cases is several independent 2D views, or this information can be converted into data structures that reflect the 3D nature of the scene more closely. Representation in these forms allow the video to be easily manipulated and displayed freely from many angles. The broad classes for techniques for the latter are as follows [2] :

- **Individual/Concatenated Views**: The simplest method to represent a 3D scene would be to leave it in its original form i.e. as individual views. If there is a one-



(a) An Image and its Depth Map [41] (b) Voxel and Octree [2]

Figure 2.15: 3D Scene Representations

to-one correspondence between the captured views and displayed views (i.e. no intermediate views need to be rendered), this scheme can be the least computationally expensive alternative. Multiple views can also be concatenated in different ways such as side-by-side or top-bottom formats. Doing so allows the use of monoscopic compression of stereoscopic or multiview video.

- **Depth Maps**: In this method, each pixel in a view has an associated depth value associated with it. Depending on the application, multiple views with their corresponding depth maps may be used. These can then be used to render intermediate views. This is a common method used to for stereoscopic (2 views) images.
- **Surface-Based Representations**: Surfaces such as polygons are used to represent parts of a 3D scene. These techniques have their basis in computer graphics. Common paradigms for this class are polygonal meshes, NURBS, point-based modeling and subdivision surfaces.
- **Volumetric Representations**: Volumetric representations simply represent the world in a 3 dimensional structure. A *voxel* or a volumetric pixel is the basic unit for representing a 3D scenes by this method. However, empty voxels may take up a large amount of space in such a representation. A more memory-efficient way is to use octrees (Figure 2.15b) for volumetric representations with varying granularity.

These classes can be grouped under Image-Based (Depth Maps) and Geometry Based (Surface-Based and Volumetric) representations. The depth information generated by converting a video into these representations can also be used for automation tasks such as object recognition and manipulation.

2.5.2.2 3D Scene Extraction form a Single Camera

Extraction of 3D scenes from a single camera video has been an area of research within computer vision for decades. It involves using monoscopic cues similar to the ones used by the human brain to extract depth information from a video. These techniques are generally grouped as Shape-From-X techniques. X being shading, texture, defocus/focus or motion. Amongst these techniques, Shape-from-Motion has been most successfully applied to real-life problems [60].

High processing requirement, poor results and dropping costs of capture equipments make such system unpopular. Therefore, the application for such technologies exists only for extending legacy systems where modifying the camera apparatus is not possible and for converting existing footage for use in 3DTV.

2.5.2.3 Human Face and Body Specific Techniques

For many scenarios, the 3D scenes are mostly composed of human face or bodies. For such cases, a priori knowledge of the structure and motion of the human face and body can be used to make the process of extracting depth more efficient [60]. These techniques usually consist of the following subtasks:

- Detection of face and facial features.
- Face motion analysis and mimic.
- Face structure capturing.
- Modeling of the human body.
- Analysis and recognition of human body motion.

Typically, these techniques use surface based representations of the human face and body to represent the obtained depth. These techniques are algorithmically complex and therefore are hard to use in a real-time scenario. Further, their human-centric nature makes their main application to be entertainment oriented.

2.5.2.4 3D Scene Extraction from Multiple Cameras

Multicamera systems capture a scene from several viewpoints at the same time as shown in figure 2.16. The remainder of this document uses the terms multiview and multicamera interchangeably. Capturing dynamic scenes with multiple cameras has a number of challenges.

As there usually isn't a one-to-one correspondence between pixels captured by a camera and the corresponding location of an object in 3D space; camera calibration is required by any system that aims to extract depth from an image. Other parameters such as exposure, white balance between cameras also need to match. Toolboxes in [61] and [66] use a moving LED as a target for multicamera calibration.

Further, if two frames from different cameras arriving at time T were actually captured at times $(T - T_1)$ and $(T - T_2)$, the two frames may be correlated incorrectly

Figure 2.16: An array of cameras for multiview capture [42]

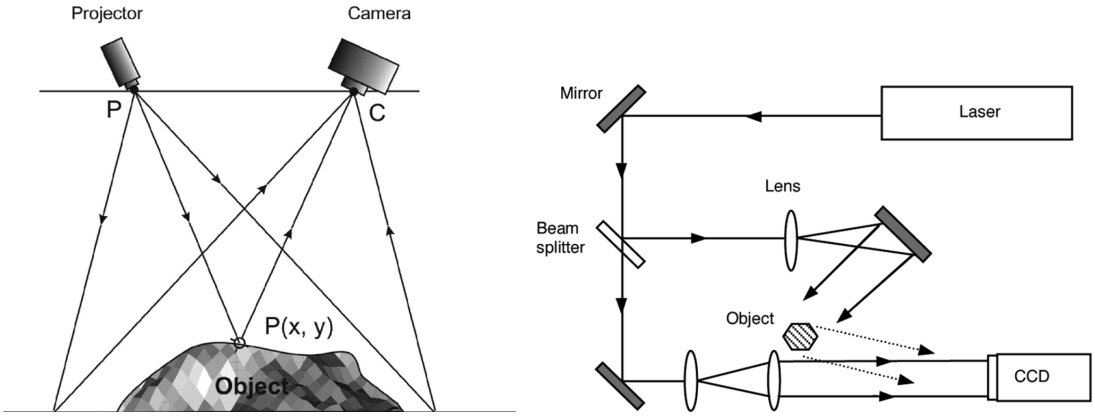
while reconstructing the 3D scene, causing artifacts. Small amount of de-synchronization can be safely ignored as human eyes are not very sensitive to rapidly moving objects but may be unacceptable for automation tasks. However, small delays Hardware and software based approaches for multicamera synchronization are proposed in [24] and [33]. Another problem is handling the large amount of data that is parallely generated by the several cameras used in such systems.

As discussed before, the multiple views can either be used directly for compression, transport and viewing or the actual 3D scene can be reconstructed from them. Reconstruction algorithms are fairly complex and therefore hard to implement in a real-time scenario. These algorithms fall into four classes [56]. Algorithms in the first class start by computing a cost function on a 3D volume and then produce a surface from this volume. For example, the voxel coloring algorithm and its variants sweep through the volume, compute costs and reconstruct voxels with costs below a threshold. The second class evolves a surface iteratively to minimize a cost function. These methods typically employ volumetric or surface-based representations. The third class of techniques computes a set of depth-maps based on disparity information between the captured views. Constraints are used to ensure consistency in between the resulting depth-maps. As mentioned before, these depth-maps can then be combined or interpolated to reconstruct a smooth 3D representation of the scene. The last class first finds a set of feature points and then fits a surface to these features.

Typically, photo-consistency is used to compute the cost function. For example, voxel coloring may by matching an assumed color for a voxel, projecting the resulting color on all the recorded views and measuring how consistent this color is with the recorded pixels. The color that minimizes the difference between actual views and projected values is then the correct color for that voxel.

2.5.2.5 Pattern Projection Techniques

This technique exploits the distortions in a light pattern projected upon the subjects of the scene. The reflection of this pattern from the objects in the scene is distorted. As



(a) Pattern Projection (b) Recording of Digital Holograms

Figure 2.17: Apparatus for 3D capture using Holograms and Pattern Projection [60]

this distortion is a function of the geometry of the objects, depth information can be extracted from the scene by processing the reflected pattern.

The advantage of such a system is that the processing requirement for extracting depth information by this method is relatively low and it can be used in real-time applications. However, this technique suffers greatly from noise and usually has a low resolution. Further, the pattern must be at a high contrast with respect to the ambient light for proper detection. It also introduces moving light into the scene which can cause eye-problems.

2.5.2.6 Holographic Techniques

Holography is the process of recording the 3D structure of an object in the form of a light interference pattern. Figure 2.17b demonstrates the apparatus for this process. A split laser beam is reflected off of an object and the interference pattern of the reflected laser with the reference wave is recorded. Recent advancements have allowed the use of CCD and CMOS image sensors to record this pattern instead of film. Colored (RGB) lasers can be used to record full colored holograms [65]. In order to successfully record the interference pattern, the angle θ between the object and the reference waves must be less than the value given by:

$$\theta_{max} = sin^{-1}\left(\frac{\lambda}{2x_p}\right) \tag{2.5}$$

Here x_p is the distance between the centers of two camera pixels and λ is the laser's wavelength [54]. This limits digital holography to only recording small objects located far from the camera. Further, the data rate of a holographic video is too high for practical transport mechanisms. Therefore, significant advancements in imaging, laser and communication are required to make digital holography of large moving objects a reality.

2.5.2.7 Time of Flight Techniques

Time of Flight(TOF) has been used for distance measurement for decades in radar and lidar systems [22]. TOF systems measure the time taken by the probing signal to travel to the target, be reflected and travel back to the receiver. The distance is then simply half of this time multiplied by the signal velocity in the propagation medium. The systems applied for 3D capture typically employ RF modulated near-infrared light from an arrray of LEDs or laser diodes, creating a uniform illumination of the object [60]. The intensity and phase of the reflected light can then be measured with an array of pixels.

Since these systems usually use a periodic signal to modulate the light, they suffer from ambiguity in the measured distance between phase shifts of 2π. Reducing modulation frequency increases the ambiguity distance at the expense of resolution. This problem is solved by using multiple modulation frequencies [17]. TOF systems are fairly low cost and usually accurate to a range in the order of millimeters. Further, these systems can operate at a very high frame rate. However, since these systems only provide intensity information, separate sensors need to be used to capture the scene in color.

2.5.3 Compression and Decompression

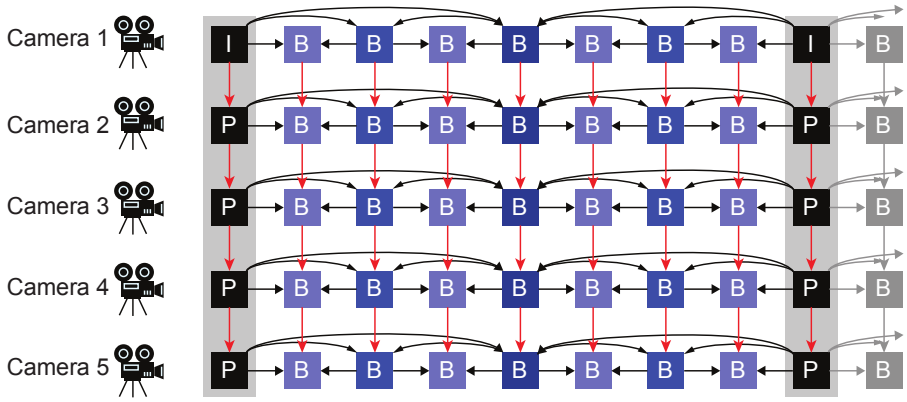
Data rates for uncompressed Stereoscopic / 3D video can be several times that of monoscopic video. Clearly, efficient compression is crucial for effective transport and storage. Further, different views of the scene are highly correlated with each other. The simplest and least computationally expensive method would be to use monoscopic compression techniques to compress 3D video by either first concatenating the several views into a single video stream or compressing each view independently. However, by doing so we are effectively wasting the correlation between views. Here, we explore these methods and others that can be applied to the several representations for stereoscopic / 3D video.

2.5.3.1 Multiview Video Coding

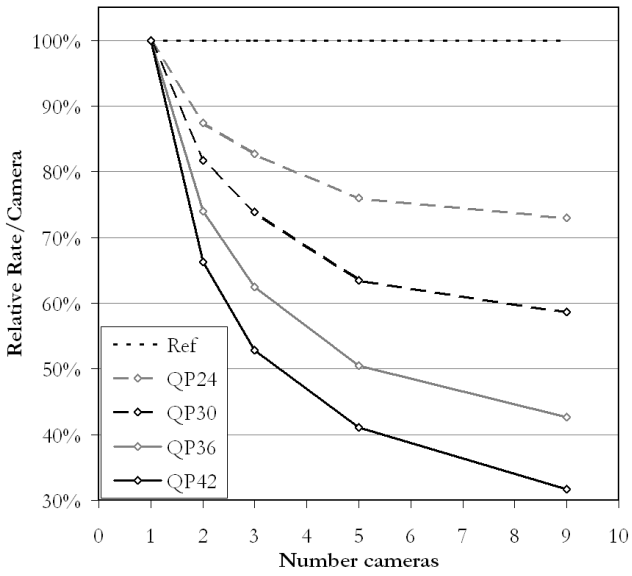
Consider a stereoscopic video. At any given time, disparity between the two views is equivalent to the dense motion field between two temporally correlated images in a video. Consequently, the same principles of motion estimation and compensation can be used to exploit inter-view correlation to achieve better compression.

However, the statistical properties of inter-view disparity vector fields and those for motion compensation [59]. Disparities are relatively larger and biased. Zero disparity correlates to an object very far from the cameras while objects close-by would cause very large disparity values. Disocclusion (content hidden behind an object in one view is visible in another and therefore cannot be predicted) is also more pronounced in between views than temporally nearby frames. Further, incorrect camera calibration

[6] Quantization Parameter (QP) affects the quality of the encoded video (higher is worse) and its size. In H.264 this can be varied from 0 to 51. Please note that the reference here is a single view video compressed with the given QP for each series.



(a) H.264/AVC Multiview coding with hierarchical B frames for temporal and inter-view prediction (red arrows) [41]

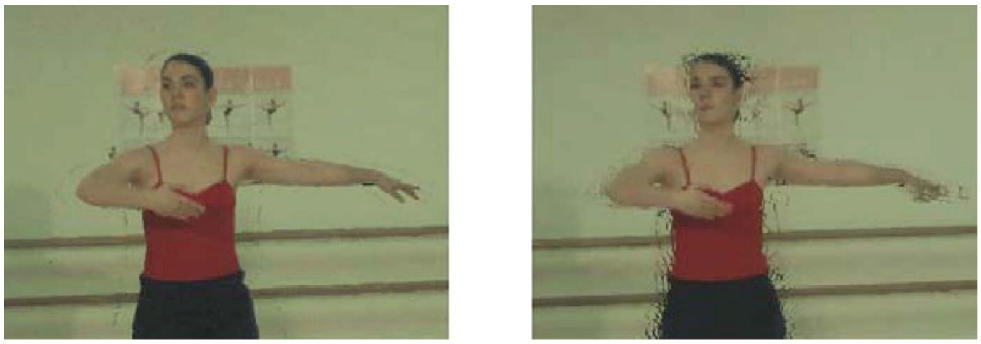


(b) Bit-rate gains for Multiview Coding [38] for different values of Quantization Parameter[6]

Figure 2.18: Multiview Video Coding and the resulting gain in bitrate.

(discussed earlier) and variation in scene lighting and reflection effects can also cause views to be more dissimilar.

Standards for such compression has already been defined in the Multiview profile of H.262 / MPEG-2 Video [21] and amendments for H.264/AVC for Multiview Video Coding (MVC) [41]. Figure 2.18a shows the process of inter-view prediction. Intra coded frames are only used in one of the views and the frames from other views are predicted from these frames. Bidirecionally predicted frames also use other bidirectionally predicted frames as reference frames.



(a) Original

(b) Reproduced from a stream with 4:1 View/Depth coding ratio

Figure 2.19: Errors caused by Depth Map Compression [41]

Peak gains with this method are reported to be 3dB PSNR[7] with corresponding bitrate savings of 50% [38]. However, [41] reports and average PSNR of 0.9dB with several benchmark video sequences. Further, as shown in Figure 2.18b, the gain increases with the number of cameras. Therefore, for systems with fewer cameras, the small gain with respect to compressing each view individually may not justify the added complexity.

A major drawback of combining temporal and inter-view prediction is its very high computational complexity, memory requirements and delay. This complexity can be reduced by only using inter-view correlation for I frames as most of the gain in MVC comes from converting the I frames to P frames [59].

2.5.3.2 View + Depth Coding

As discussed before, depth information can be derived from multicamera systems or recored directly with cameras using techniques like pattern projection or time of flight measurement. The obtained depth information can be compressed as a video stream as brightness values. In case of a stereo camera pair, one view and depth is compressed. For multiview, depth infgormation is associated with each view.

Depth data is usually encoded on to an 8 bit luma channel and chrominance values are set to a constant value, effectively encoding it as a monochrome video. Because of the characteristics of human perception, inverse depth $1/z$ is quantized uniformly, allowing finer depth resolution for nearby objects [41]. For correct reproduction, the received depth needs to be inverted before display. MPEG has specified a corresponding container format "ISO/IEC 23002-3 Representaion of Auxillary Video and Supplemental Information", also called MPEG-3 Part 3, for video + depth data [10]. Further, H.264/AVC also allows the encoder to add depth information via its auxiliary picture syntax.

[7]Peak Signal to Noise Ratio is a common measure of codec quality. It is used to compare how close the codec is to the original at the same bitrate as the reference codec.

As with disparity vectors, the depth data is statistically different from normal video. Color video typically contains detailed texture information. On the other hand, depth data usually consists of large homogeneous regions of show-changing values. Also, object boundaries typically contain abrupt changes in depth values. Consequently, depth data is primarily composed of very low and very high frequencies. Higher frequencies are removed at higher compression rates. For normal video, this only leads to slight degradation of visual quality of the video. However, as shown in Figure 2.19, the effect on depth information is much more severe. While artifacts caused by removing high frequencies from normal image or video is simply spurious objects; similar artifacts in depth information are near / far depth objects floating on a far / near background. Clearly, sharp edges of object depth and discontinuities in depth are highly visible and can be very discomforting to the viewer.

As mentioned earlier, stereoscopic video is encoded as a single view + depth data. In this case, an additional problem is with objects occluded in transmitted view being visible in the omitted view. This means parts of background occluded behind a foreground from the transmitted view show up as blank areas in the reproduced video. This can usually be corrected by filling such areas by spatially neighboring content.

2.5.3.3 Geometry Coding

This class of algorithms is used to compress 3D content represented with surface-based and volumetric methods. These algorithms have their origins in computer graphics where it is required to compresses 3D meshes to reduce the data that needed to be transferred from the main memory to the graphics card. These algorithms have now been applied for compression of 3D video. Considerable research has been done in this area and covering the multitude of algorithms is outside the scope of this report. Therefore, we limit ourselves to the principles behind geometry coding.

As mentioned before, surface-based representations use polygons as basic blocks. Position and connectivity information about these polygons is required to construct a surface. This is called a 3D mesh. Algorithms for compressing meshes can be divided into two categories: *single rate encoders* and *progressive encoders*. Single rate encoders compress the mesh into a single bit stream containing both the connectivity and position information of the mesh. Progressive encoders first simplifies the mesh into a base representation that contains fewer polygons. It then compresses the base mesh into a bit stream. It then encodes a sequence of operations that undo these simplifications. This method allows the decoder to recreate the simplified mesh with the smaller base mesh and then build upon it progressively as new information arrives. This is not to be confused with multi resolution meshes, where independent meshes of increasing resolution may be used.

Single rate encoders traverse the mesh, generating a symbol for each triangle. These symbols are then entropy coded [46]. Further, prediction is used to concentrate the

statistical distribution around small values. In essence, each time a new vertex is visited by the encoder, its coordinates are predicted by those of the previously visited vertex. The difference between the actual position and the predicted position is then entropy coded. Since these errors are generally small values, fewer symbols are needed for entropy coding, causing an increase in coding efficiency.

Progressive encoders first simplify the mesh, reducing the number of triangles used to represent the surface. The resulting base mesh is then compressed using a single rate encoder. The sequence of simplification used to create the base mesh are then reversed by encoding opposite refinement operations. In an animation sequence, successive meshes are encoded as transformations on previous meshes. In analogy to video compression, independently encoded meshes are called I meshes while predicted meshes are called P meshes. The MPEG-4 standard supports mesh compression in the form of 3DMC tools [1].

Volumetric representations are commonly compressed using Octrees. First, the entire volume is divided into octants. For each octant, a set of motion vectors are computed. Then, an error measure is computed to compare the individually calculated motion from the cumulative motion of the octant. If this error measure is less than a threshold, the octant is encoded. Otherwise, the octant is subdivided again[46]. The algorithm recursively creates octants and computes error measures until the error for the derived octant falls below the threshold. The algorithm stops when the entire volume has been traversed. Needless to say, geometric compression and decompression algorithms in general require a considerable amount of processing power.

2.6 Factors of Human Comfort

Stereoscopic / 3D remote vision has considerable advantage over standard monoscopic remote vision. For example for teleoperation applications, [9] reports 10 times improvement in task completion time for stereoscopic over monoscopic systems. Another study [8] claims that the "overall psychological impact of a 3-D screen is equal to flat images twice their size". As with anything that improves our lives, there is a catch. A number of mismatches between 3D displays and the human system for depth perception leads to decreased viewer comfort. Symptoms exhibited after using 3D displays include headache, disorientation, nausea and breakdown of 3D perceived from the display.

As mentioned before, a major problem is the accommodation-vergence mismatch. This mismatch is depicted in Figure 2.12. While viewing an object on a 3D display, our eyes are always focused on the screen but in order to perceive an object, they have to converge to its apparent depth. This conflict of depth cues can lead to serious user discomfort. This discomfort grows with object depth[8] perceived with stereopsis. As a result, many studies define the so called '*working areas*' as the region of binocular disparity where users are found to be comfortable. According to [9], the

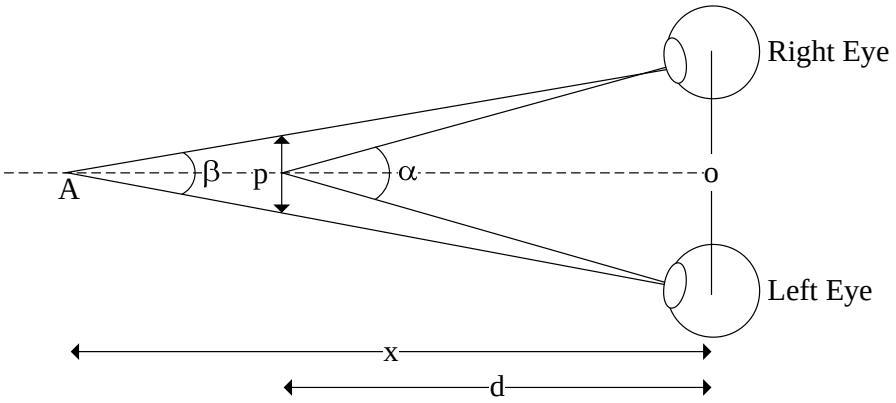


Figure 2.20: Angular Disparity

most conservative estimates put the limits on crossed angular disparity[8] to 0.45° and 0.4° for uncrossed angular disparity. Another experiment puts the limits between 4° – 7° for crossed disparity and 9° – 12° for uncrossed disparity and claims that this variation in the range stems from factors like lighting conditions, contrast and proper image blending. Some research suggests dynamically adjusting disparity by adjusting the angle of cameras. This is done by converging or diverging the cameras in a manner similar to our own eyes to bring the prominent objects at near-zero depth. This is said to improve user comfort and task completion times [9].

To make sense of the above angular values, the mathematics of angular disparity is described here. Since the disparity of objects depicted on a screen varies with the viewing distance and the distance between his/her eyes, angular disparity is used as a measure of disparity. Lets say the viewer is located at a distance d from the screen and an object B is is shown at the screen such that the viewer perceives the object at a depth of x as shown in Figure 2.20. If the distance between the eyes (called inter-ocular distance) is o then:

$$tan\left(\frac{\alpha}{2}\right) = \frac{o}{2d} \tag{2.6}$$

$$tan\left(\frac{\beta}{2}\right) = \frac{o}{2x} \tag{2.7}$$

Here α and β are visual angles. Therefore, the angular difference θ perceived by one eye is:

$$\theta = \frac{\alpha - \beta}{2} = tan^{-1}\left(\frac{o}{2d}\right) - tan^{-1}\left(\frac{o}{2x}\right) \tag{2.8}$$

And the resulting angular disparity μ is:

$$|\mu| = |2\theta| = \left|2\left(tan^{-1}\left(\frac{o}{2d}\right) - tan^{-1}\left(\frac{o}{2x}\right)\right)\right| \tag{2.9}$$

[8]Crossed disparity is the disparity observed when the object is perceived in front of the screen. In this case, the left and right images of the same object cross each other.

However, the screen depicts linear disparity p is given by:

$$p = 2(x-d)tan\left(\frac{\beta}{2}\right) \tag{2.10}$$

$$|p| = o\left|1-\frac{d}{x}\right| \tag{2.11}$$

Other than the accommodation-vergence rivalry, vertical disparity and crosstalk are known to be the most significant factors that determine user comfort [8]. Kooi and Toet at TNO [28] have evaluated the effect of several asymmetries between the left and right views. Some of their key findings are:

- Horizontal shift should be less than 1.14° of visual angle.
- Vertical shift should be less than 0.57° of visual angle.
- Difference in contrast (often caused by polarizing filters) between the left-right views should not exceed 25%.
- Relative magnification of one view with respect to another should be less than 2.5%.
- Crosstalk between the left-right views should be less than 5%.
- People with reduced binocular vision find misalignments such as horizontal and vertical shifts less troublesome and the quality of binocular vision is determined by the poorer eye.
- Discomfort increases with disparity. This is because effects like crosstalk get more pronounced with disparity.

Another cause of user discomfort is the keystone effect produced when the cameras are converged or '*toed-in*'. Although this configuration helps control depth in the scene, the space seen by cameras is warped. The perceived space is compressed near the center of the scene and expanded at the outer edges. This is a common cause of vertical shift. Lens distortions are also found to commonly cause mismatch between the left and right views. A mathematical model of image distortions in stereoscopic systems is presented by Woods et. al. in [64]. They also suggest that increased exposure to stereoscopic displays seems to improve user comfort [64]. This is not to be confused by prolonged usage which can lead to increased discomfort over time for systems that exceed the above recommendations. Another comfort factor worth mentioning is screen flicker. The frequency at which screen flicker becomes just visible (critical flicker frequency) increases with with luminance [8]. Another influence on critical flicker frequency is the screen's surroundings.

In conclusion, accommodation-vergence rivalry is the primary cause of user discomfort. Unless volumetric or holographic systems are used, this problem can only be circumvented and not solved. However, as discussed earlier, these systems require significant improvements in current technologies before they can become practical. Other causes of discomfort are generally caused by asymmetries between the left and right views. These can be caused during the capture or display processes. Humans are resilient to a certain degree of these asymmetries. A system that remains within these bounds can be be used without the user experiencing discomfort.

3 Design and Implementation

"The most exciting phrase to hear in science, the one that heralds new discoveries, is not 'Eureka!' but 'That's funny...' " – Isaac Asimov

The previous chapter was focused on exploring various technologies to develop the targeted remote vision system. We also discussed several aspects of teleoperation and human perception that shed light on the desired characteristics of the targeted system. In this chapter, these characteristics are crystallized and merits of these technologies are weighed against these requirements. This results in a set of high-level design decisions which are followed by lower-level design decisions coupled with implementation specific details they are derived from.

3.1 Requirements

In order to make informed design choices, it is important to understand the desired characteristics of the targeted system. The research presented in Chapter 2 provided insight into these requirements which are crystallized here:

- **Delay:** Delay has a significant effect on the performance of the operator. As suggested by Smed et. al. [58], human control of a system is fluid when delay is lower than 200ms; higher delays cause it to become more cognizant. However, this value is more relevant for computer gaming where faster responding individuals tend to dominate the game. Since teleoperation is typically performed with unnatural controls, the operator needs to think before they perform even the simplest task. Therefore, teleoperation is not intuitive but cognizant to begin with. Consequently, this requirement can be relaxed in our context.
- **Jitter:** Jitter, or variation in delay, can also affect operator performance significantly. Since camera hardware is inherently periodic, jitter can be caused by software or the network. Therefore, care should be taken at software level to ensure timely behavior. Further, features in streaming and display software need to be chosen to avoid network jitter along with appropriate network protocols.
- **Bandwidth:** Minimizing the video data rate with compression relaxes the required network capabilities for the system to work. It also reduces the network delay. However, this adds processing delay for compression and decompression and reduces video quality. Therefore, bandwidth vs video quality is a tradeoff that needs to be made carefully.
- **Binocular Disparity:** As discussed before, accommodation-vergence rivalry is a major cause of discomfort for viewers. Therefore, binocular disparity needs to be

minimized. More importantly, out-of-screen or crossed disparity should be minimized as this is known to be more discomforting. Suggested limits for binocular disparity lie between $4^\circ - 7^\circ$ for crossed disparity and $9^\circ - 12^\circ$ for uncrossed disparity [9]. Disparity can be controlled at different levels. If the chosen representation format encodes depth separately, it can be done at the rendering machine, otherwise this needs to be done by manipulating the camera configuration.

- **Camera Convergence:** The camera convergence or the degree by which the cameras are *'toed-in'* should be adjusted to a convergence distance such that fewer objects are displayed with crossed disparity and the objects to be manipulated (therefore the objects the operator looks at most) are at near zero disparity. However, toed-in camera configuration also causes *keystone distortion*. Keystone distortion is perceived by the user as a warping of space, and more importantly as vertical shift at the sides. Consequently, there is a trade-off between minimizing discomfort caused by accommodation-vergence conflict and vertical shift.

- **Crosstalk:** Crosstalk between the left and right views is a common problem with several display types. The level of crosstalk should not exceed 5% [28]. This needs to be taken into consideration while selecting a dsiplay technology.

- **Contrast Difference:** Difference in contrast between the left and right views can be caused by difference in camera properties and scene lighting. In many teleoperation scenarios, scene lighting cannot be controlled but camera features like automatic exposure control should be synchronized or turned off. Contrast difference can also be caused by the characteristics of a display. For example, displays based on linear polarization will have varying contrast in the perceived left and right views for different angles of user's head with respect to the display.

- **Left-Right View Misalignment:** Misalignments, between left-right views can also cause user discomfort. The most important of these misalignments is vertical shift which should be limited to 0.57° of visual angle [28]. This limits the acceptable error range in camera positioning. Recommendations for other, less significant left-right view asymmetries can be found in section 2.6.

- **Video Quality:** It is hard to quantize the quality of video. Factors such as video resolution and compression artifacts affect the video quality. Increasing the compression level decreases bandwidth, network and processing latency at the expense of increased compression artifacts. Decreased delay would typically improve the operator's performance but decrease in video quality would make it worse. Therefore, adjusting the compression level to optimize operator performance is an important task.

- **Immersion:** It is important for the operator to have a more immersive experience of 'being there' and his/her performance is likely to improve with the quality of immersion. For example, allowing for depth perception is known to increase operator performance. However, immersion does not just stop at depth perception. Several other characteristics of the display can affect the degree of immersion. These are discussed further in the rest of this chapter.

- **Extensibility:** As discussed later, highly immersive display systems can be very costly. It is economically viable to start with a display system with low initial cost. An ideal display system would be one that can then be extended for higher immersion with little additional effort.

- **Cost:** As with any other system, cost is a major factor for the suitability of the remote vision system for its various applications. Most importantly, remote assistance can be provided to multiple individuals from a single teleoperation machine. This many-to-one relationship means that the 'proxy' system should be low cost while a costlier teleoperation system can be used.

- **Development Effort:** The system must be developed while best utilizing the currently available hardware and software packages such that the overall development effort is little. Making use of available software packages also means that the target system benefits from future improvements in those packages with little effort.

3.2 Delay Analysis

As discussed before, delay plays a significant role in operator performance. Therefore, it is important to discover the parts of the system that affect delay wherever possible. Assuming the cameras operate a t 15 frames per second, the worst case latency between an event happening in the real world and being registered in the form of a frame is $1/15 = 66.67ms$. Correspondingly, typical displays refresh (show new information) at a rate of 60Hz, i.e. an additional latency of $1/60 = 16.67ms$. These capture and display delays of the hardware devices alone constitutes a significant amount of delay in itself.

Network delay can also play a significant role. A typical network consists of several switches and routers between network end-points. In this case, the two end-points are the remote 'proxy' machine and the teleoperation machine. Network latency mainly consists of the time spent by packets in these network devices. For example, the task of a network switch is to store and forward packets. They do so by use of input and output queues at each port. When a packet arrives from a port, it is placed in the input queue corresponding to that port, after looking up the destination address of that packet, it is placed int the output queue of the destination port. The time taken to do this depends on the packet's priority and the number of older packets in the queue. Other factors include the memory latency of the switch. For the highest priority packet on 100Mb/s Ethernet, given that there are 23 prior highest priority packets in the queue, the total time spent by a 1000 byte packet in the switch is approximately 2.06ms [14]. The time taken to transport a combined frame of from two cameras, at a resolution of 1024x768 each with 16 bits for each pixel is:

$$\frac{1024 * 768 * 16 * 2}{100Mb/s} + 2.06ms = 242.06ms$$

At the resolution of 512x384 each, this is reduced to 62.06ms which is still considerably

Technology	Advantages	Disadvantages
Autostereoscopic (Philips Lenticular Screen)	+ No need to wear glasses	– View + Depth Format Required – Fixed Viewing Zones, 3D effect breaks down elsewhere – Expensive – No support for Extensibility
Poalrized (iZ3D and Zalman)	+ Cheap glasses: easy to add users	– Contrast varies with angle of user's head for linear polarization (iZ3D)
	+ Cheap	– Imperfect polarization at display often causes crosstalk – No support for extensibility
Field Sequential (nVidia 3D Vision)	+ Support for extensibility (up to 9 screens)	– Synchronization issues at high frequencies can cause flicker and crosstalk
	+ Independent left-right view buffers	– Glasses are expensive compared to polarization filter glasses
	+ Cheap	– Bright areas have more pronounced crosstalk

Table 3.1: Comparison of commercially available 3D displays

large. Further, this delay will increase with the number of switches and routers between the end-points.

What remains is the delay caused by processing of this data at the remote and teleoperation machine. This may include transport delays between hardware and software, transformation to a representation format, compression, decompression and color space conversion. However, 145.5ms are already lost during capture, over the network and display. Further, the data rate for internet is typically lower than Ethernet. By compressing the frame data, the network delay can be reduced significantly. As discussed later, compression has indeed been used for this system.

3.3 Display

The choice of display can directly affect operator performance. Not only is it important to select a display technology that has good technical characteristics such as low crosstalk, but also to maximize the degree of immersion that can be achieved with these displays. The 'suspension of disbelief' (what you see on a screen is not real), decreases with increasing size of the screen. A popular example in literature is the CAVE (CAVE Automatic Virtual Environment) system [13]. A CAVE is essentially a room where the user is surrounded by stereoscopic projections on all walls, ceiling and floor.

In order to create an immersive experience for teleoperation, we intend to allow the user to look around at his/her proxy's surroundings. The user's head can then be tracked with the help of one of the several tracking technologies discussed in [52]. By capturing the part of the proxy's surroundings that corresponds to the user's head orientation, and displaying it at the corresponding location, it is possible to create a highly immersive teleoperation system. This would involve either using large displays or an array of displays.

Projectors can be used for creating a large display for look-around capability. However, looking around would not feel natural with a flat screen in which case the space around the user would seem warped in a way opposite to the keystone effect. Instead, a dome like structure of display curved around the user is required. The author is not aware of a projector based stereoscopic display capable of projecting on a curved screen. Moreover, as discussed before, projector based stereoscopic displays use linear polarization which has a disadvantage of contrast variation in the left and right views depending on the orientation of the user's head. Therefore, an array of screens is the better alternative. However, this adds extensibility requirements on the display system. An ideal system would be one in which it is possible to start with one or more screens and allow for the system to be extended to several screens.

Table 3.1 compares the commercially available 3D displays. The nVidia 3D vision system offers the ability to either play a single video extended across as many as 9 screens simultaneously or to play several separate videos on each screen. This is achieved by coupling several graphics cards together. A single nVidia graphics card can play a stereoscopic video on 2 screens. by using OpenGL and nVidia Quadro cards, separate video buffers for the left and right eye views can be written to independently. This means any video representation format can be used with this display system by rendering two different views. These qualities led us to use the nVidia 3D vision system for our remote vision system. As we shall see, the current implementation only uses one screen. Extending the system is left as future work but can be achieved with the present hardware.

3.4 Capture

The available capture hardware for this project were generic cameras. In order to surround the operator with video of the proxy's surroundings, it is required to have a camera configuration that can capture video from several angles. This kind of configuration is the opposite of a typical multiview capturing system in which multiple cameras are oriented towards the same set of objects as shown in Figure 2.16. This means that the user cannot perceive motion parallax with such a configuration. However this is not a significant issue as the ability to look around is more important than motion parallax in a teleoperation scenario. However, such a system would pose very heavy requirements in terms of size, processing power and bandwidth. This would significantly increase the delay and cost of the system.

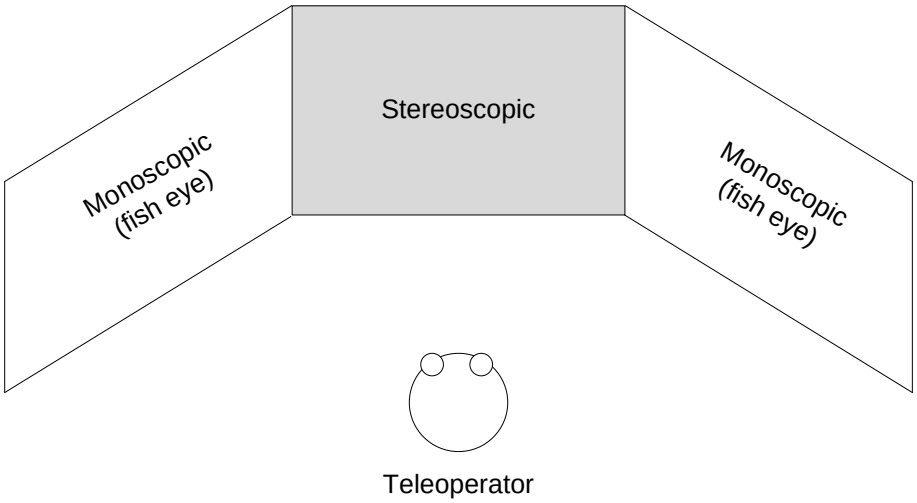


Figure 3.1: Extending current stereoscopic remote vision system with a single fish-eye camera and 2 additional screens for peripheral vision

For the above mentioned reasons, we opted for a stereoscopic approach instead of multiview. In order to provide the viewer with a wider field of view, it is possible to use a fish-eye lens with a single additional camera. Fish-eye lenses can increase the field-of-view of cameras to up to 180°. This would allow a wide field-of-view with the addition of only one camera. Therefore, stereoscopic video would be available in only one of the screens and the rest would display a monoscopic view of the proxy's surroundings. As the user looks around, the orientation of his/her head can be used to change the orientation of the stereoscopic cameras with the help of actuators in a manner similar to the operator's neck movement. Thus, stereoscopic cameras coupled with head pose detection can be used to provide nearly the same level of immersion with much simpler hardware and lower delay. The current implementation only consists of two cameras, the addition of the fish-eye camera is left as future work.

With stereoscopic cameras, the choice of representation format is between leaving them in their original form as individual views, concatenating the views into a single video stream, or converting to View + Depth format. A comparison of these formats is presented in Table 3.2. The View + Depth format which requires implementation of stereo matching algorithm and entails added delay because of the additional processing required before the video can be compressed. Another disadvantage of using View + Depth format is caused by occlusion. Consider a scenario where the View + Depth format consists of the scene as seen by the left camera and the depth map. Parts of the scene that are visible to the right camera and occluded from from the left view (an object behind another) are not transmitted and the viewer sees this as a blank in the right view. One workaround is to fill the resulting blank areas with neighboring content but the left right views are still inconsistent which can confuse the operator and possibly decrease his/her

[1] Computational complexity of calculating depth maps varies from as much as $O(N^N)$ to $O(NDlogD)$ where N is the number of pixels and D is the range of disparity in pixels [12].

Format	Advantages	Disadvantages
Individual Views	+ No manipulation required at capture	– Each view needs to be compressed individually
	+ Higher compression ratio than concatenation possible	– Higher data rate than View + Depth
Concatenated	+ Only concatenation required	– Higher data rate than View + Depth
	+ Simple monoscopic compression	
View + Depth	+ Data rate 60% to 70% of others [59]	– Disturbing artifacts from depth map compression
		– Depth needs to be computed[1]
		– Objects occluded from transmitted view can cause blank areas

Table 3.2: Comparison of stereoscopic representation format

performance. Therefore, it was decided to not use the View + Depth format. The choice between the other two options (individual vs. concatenated views) is discussed later in Section 3.5, as this choice is influenced by compression.

3.5 Compression

In the previous sections, many 3D representation formats were ruled out. The choice now lies between compressing the two views individually or as a single concatenated monoscopic video. As mentioned before, the correlation between the left-right views is akin to the temporal correlation between neighboring frames of a monoscopic video. Therefore, similar techniques can be used to compress one view with respect to the other. However, the left-right views are much more dissimilar than temporally neighboring frames, making it harder to exploit this correlation. Further, as shown in Figure 2.18b, the gain by compressing different views of a scene with respect to another increases with the number of views. Most importantly, motion estimation is the most demanding algorithm for a video encoder and takes up 60% – 80% of the total computation time [29]. Conseqently, inter view disparity estimation can be expected to take up significant amount of computation time. Therefore, we chose not to implement inter-view compression. Further, this allows us to use concatenated views with monoscopic compression techniques.

As mentioned before, motion estimation requires a large amount of processing time. In order to exploit temporal correlation with motion estimation and compensation, the encoder must accumulate a certain number of frames, estimate the motion from past and future frames and then transmit the resulting group of frames. Therefore, a few frames must be buffered before any frame is transmitted. An investigation into

Codec	Scene Composition	Configuration	Bitrate (kb/s)
MPEG-4	Still	Default	3063
MPEG-4	Still	Only I-frames	9549
MPEG-4	Motion	Default	6136
MPEG-4	Motion	Only I-frames	7702
MotionJPEG	Still		11293
MotionJPEG	Motion		9369

Table 3.3: Comparison of codecs for various configurations and scenes

the bitrates for different codecs with different scenarios. The resolution used was 640x480 pixels at 22 frames per second. FFmpeg package was used on an Ubuntu 9.10 machine to encode and transmit the video using the RTP protocol while mplayer was used to view the video live on the same machine. The encoder was configured to maintain the same quality of output video as the input with variable bitrate i.e. the encoder varied the quantization parameter according on the complexity of the scene to maintain the same quality as input. Further, the encoder was configured to use only Intra frames. This means that no motion estimation was used and each frame was compressed independently, or to use the default configuration with P and B frames. The MotionJPEG[2] codec which compresses each frame individually using JPEG compression was also used for comparison.

The results are present in Table 3.3. The bitrate for MPEG-4 with Intra frames and MotionJPEG does not change much between motion and still scenes. In fact it was found to decrease because the compositional complexity of the scene decreased with the moving object present in front of the camera. On the other hand, the bitrate for default configuration doubles with motion. On an average, the bitrate for the default MPEG-4 configuration was approximately 53.3% of that for MPEG-4 with Intra frames and 44.5% of MotionJPEG. Finally, we decided to use the MPEG-4 codec with Intra frames only. This has a bitrate advantage over MotionJPEG. Most importantly it is a standard codec and by using libraries or software that supports the standards, we can easily reconfigure the system to use different MPEG-4 configurations. As mentioned before, using Intra frames reduces processing requirement and frame buffering delay at the expense of bandwidth. The increase in bandwidth was found not to affect delay as much as the corresponding buffering delay.

3.6 Transport

One important decision when designing a system that uses computer networks for communication is that between TCP and UDP protocols. The TCP protocol offers

[2] Please note that MotionJPEG codec is not specified by any standard so the specifics vary greatly between software.

reliability, congestion control and in-order delivery. However, it does so at the cost of speed. In order to provide congestion control, TCP protocols increases speed until it observes a collision upon which it reduces speed. It also requires that every packet be acknowledged. Consequently, upon collision, the complete stream must suffer a delay until an erroneous packet is retransmitted. UDP on the other hand, simply sends packet with no congestion control and packet ordering is not preserved. It does offer a checksum to ensure that the contents of the packet are not corrupted. Video streams for teleoperation must suffer the minimum amount of delay possible. Additionally, many video applications are error resilient, it may be possible to decode a frame if some parts are corrupted. Further, a frame may be skipped over if a newer frame is already available.

Thus, real-time media systems[3] require what is called application-level framing, which implies that only the application has sufficient knowledge to decide what to do when a network error occurs; it may chose to ignore the error, ask for a retransmission, or use a lower-fidelity copy of the same data [49]. Thus, the transport protocol must expose as much information to the application as possible. This is at odds with the design of TCP protocol. Therefore, the UDP protocol is more suitable than TCP. For the same reasons, it is commonly used for media streaming.

However, UDP lacks some features that are desirable for real-time media applications. Sequence and timing information is important for real-time media applications to know when a certain packet was transmitted in order to play the video in correct order and to detect out-of-order or lost packets. For example, the decoder can use the timestamp to ensure smooth playback of the video; thereby compensating for network jitter. The Real Time Protocol (RTP) implements these features on top of UDP. It provides packet sequencing which can be used to detect out of order or lost packets. In addition, each RTP packet contains a timestamp which corresponds to the sampling instant of the first byte of data in that packet. It also contains a marker bit that applications can set for various purposes; for example to indicate that this is the last packet in the video frame, so the receiver can start decoding. Therefore, we chose to use RTP over the UDP protocol for our system.

3.7 High-Level System Architecture

So far we have selected a set of technologies to use:

- Display: nVidia 3D Vision
- Capture: Stereoscopic, Concatenated
- Compression: MPEG-4 with Intra frames
- Transport: RTP over UDP

The rest of this chapter discusses low-level design and implementation of the remote vision system. Before doing so, it is important to establish the high-level flow of video

[3] In this context, real time media systems are ones in which the receiver plays the media as it is being sent rather than storing it for later use.

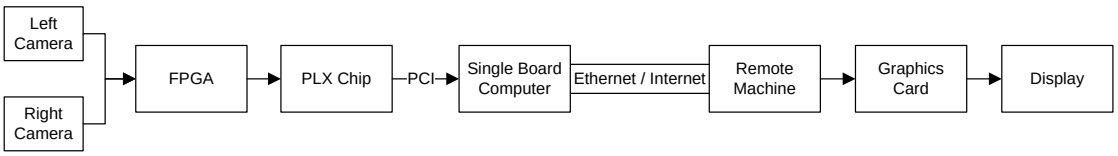


Figure 3.2: The flow of video from key hardware elements

through the remote vision toolchain. This is shown in Figure 3.2. The video is captured with cameras capable of capturing in raw Bayer format at a resolution of 2500x1600 at up to 20 frames per second. The cameras allow configuration of several parameters such as resolution, frame rate, white balance etc. via an I^2C interface. The video flows from the cameras to a Xilinx Virtex 5 FPGA where it can be converted into 16 bit or 32 bit RGB formats. Further, the two video streams from the cameras can be concatenated at the FPGA. The FPGA interacts with the PLX 9056 chip via a local bus interface[4]. The video is then available to the Single Board Computer (SBC) via the PCI bus which contains a Celelron M 1GHz processor and 512MB of DDR2 memory. The system runs the Ubuntu 9.10 operating system which uses the Linux kernel version 2.6.31. In addition the SBC supports communication with the 33MHz PC/104 (PCI) bus and 10/100 Base-T Ethernet. Here, the video is processed and transferred via Ethernet to the remote machine that runs Windows XP. On the remote machine, the teleoperator can see the stereoscopic video on a 120Hz LCD monitor with the help of an nVidia Quadro FX 3700 graphics card and LCD shutter glasses.

3.8 Cameras and FPGA

Figure 3.3 presents a high level view of the cameras and the video processing done at the Virtex 5 FGPA before it reaches the SBC. The video from the cameras is output in a manufacturer specific serial format. This is converted to 8 bit parallel format, thereby reducing the data rate by a factor of 8. Also, Hsync and Vsync signals are added before the video is available to the FPGA. The Hsync signal is high when a line of pixels is being transmitted. It stays low for a short period where no pixel data is transferred. Likewise, the Vsync signal is high during the duration of a meaningful frame and low for a definite period between frames. Aside from synchronization, the blank periods also serve other purposes. This blank period allows some time for the cameras to integrate the pixel values from the sensors and perform Analog to Digital conversion. It also allows for the receiving system to process the line or frame before new data is available. These periods depend on the frame rate the cameras are configured for. The camera adapt module principally preforms clock space conversion between the FPGA and camera clock. As mentioned before, the cameras can be configured via I^2C to output the video at different resolutions. However, the output video is always padded with dummy pixels to the highest resolution of 2500x1600. The cameras were always configured to keep the frame size at a 4:3 width to height

[4] Please note that although the author was actively involved in the design and testing of the VHDL code for the FPGA, the author of this code was Hans Kanters (Philips Apptech)

ratio. The dummy pixels are removed at the crop module. Further, the positions of the cameras were adjusted to keep the horizontal and vertical misalignments to minimum.

The FPGA can be configured to forward the video coming from the cameras in either its original Bayer format (discussed in Chapter 2) or in 32 or 16 bit RGB formats. The 32 bit RGB format consists of 8 bits for each color channel and an additional 8 bits for the Alpha or transparency mask. In our case, this mask is always set to a constant value. The 16 bit RGB format is called RGB565 which consists of 5 bits each for the Red and Blue channels and 6 bits for Green. This means it represents the Green channel with double the resolution of Red and Blue channels. As discussed before, this is because humans are much more sensitive to the Green colour. Further, it can also be configured to output the video for either the left or the right view or both in concatenated form. Concatenation of images is performed in side-by-side format rather than top-bottom format as top-bottom would require a complete frame to be buffered at the FPGA or capture of left-right frames one after the other. Thereby reducing the frame rate by half. Further, the resulting temporal variation between the left-right frames could also cause motion sickness. The conversion from Bayer to RGB formats is essentially a demosaicing process. The available processing power of the SBC limited the resolution that could be compressed and transmitted. This led us to use an averaging algorithm that combines 4 pixels in the incoming Bayer format into one RGB pixel, thereby down sampling the incoming video. As a result, the maximum resolution available at the SBC was 1024x768 pixels for each camera.

The processed video is then buffered at a FIFO with a depth of 16384 elements of 32 bits each. This buffering prevents data from being lost in case of differences in the timing of transfer requests from software side. Further, the FPGA also detects a FIFO overflow and indicates this error in its error register. Upon initiation of a subsequent transfer, it clears up the FIFO and holds off until the next frame arrives. The PLX chip communicates with the FPGA using a manufacturer specific local bus. The local bus IO module is responsible for interacting with the PLX chip. It also exposes status, error and configuration registers for its internal modules at different addresses accessible at the SBC via the PLX driver. The author was actively involved in gathering requirements, design, testing and debugging of the FPGA code. The actual development was carried out by another team member. Therefore, the details of this system are abstracted from this discussion. Instead, later sections focus on the rest of the toolchain.

3.9 Software Toolchain

The video is transferred from the FPGA to the SBC via the PCI bus. To enable this, the FPGA board contains a PLX 9056 chip. PLX also provide corresponding drivers for the Linux and Windows operating systems. A corresponding SDK is also provided. Therefore, a user space application can obtain the video from the cameras. The rest of the toolchain consists of compression, transport (streaming over RTP), decompression and display. One advantage of choosing to concatenate the left and right camera views is that we can use monoscopic compression techniques. Several software packages

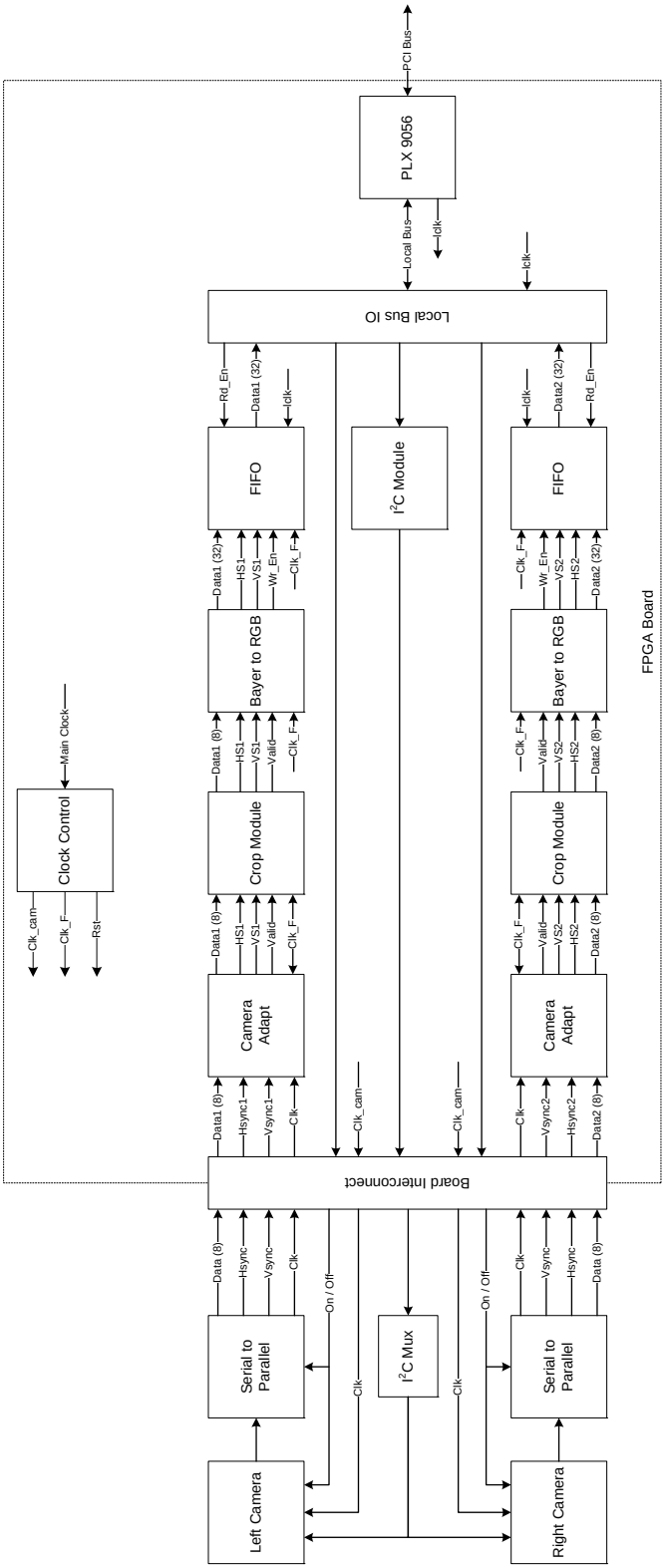


Figure 3.3: High level schematic of the remote vision system: Cameras and FPGA

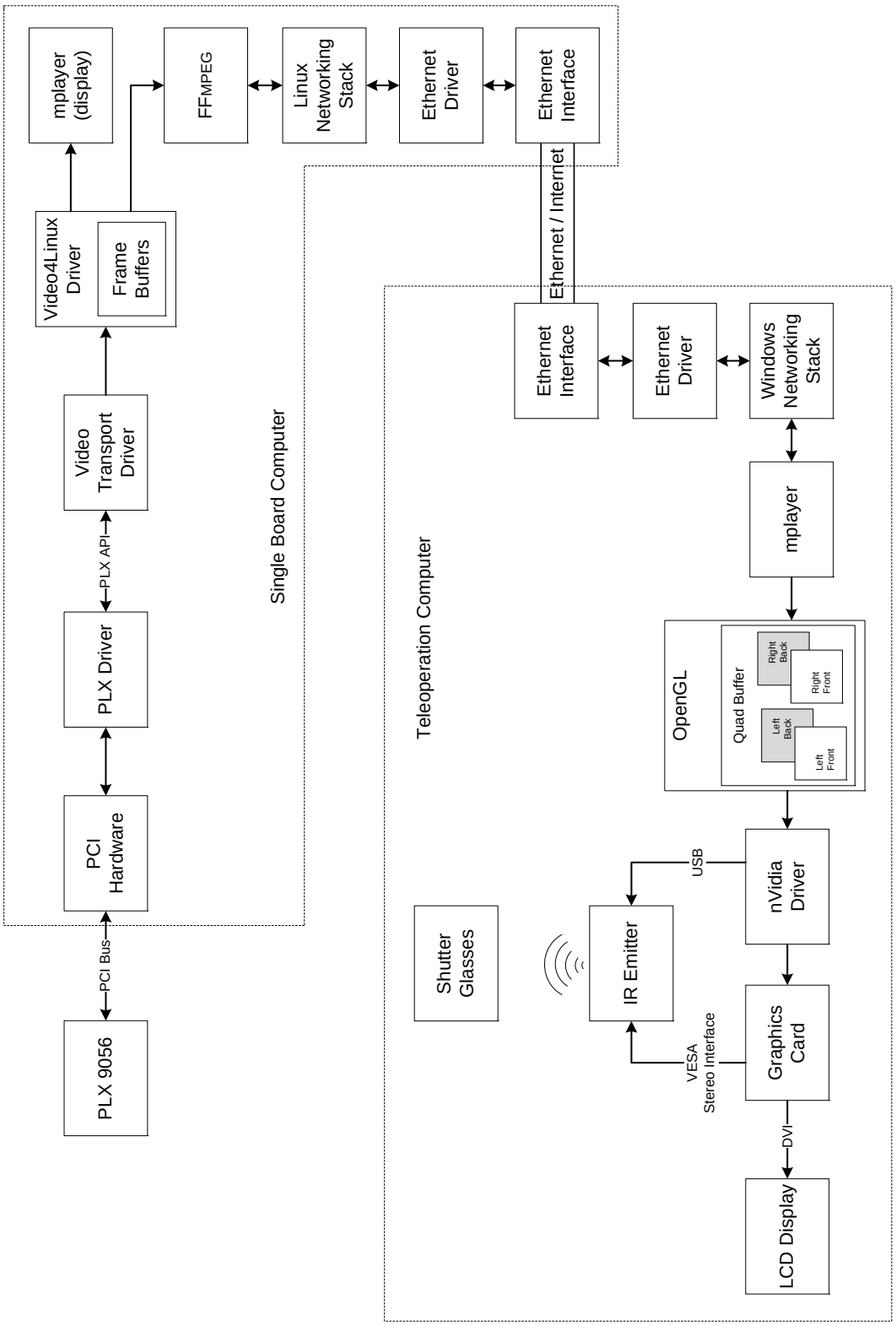


Figure 3.4: High level schematic of the remote vision system: SBC and Teleoperation Computer

Package	(De)Compr.	Server	Client	Display	License	Library / Binary
mplayer	✓ (mencoder)		✓	Monoscopic; Stereoscopic	GPL	Binary
VLC	✓	✓	✓	Monoscopic	GPL	Binary
FFmpeg	✓	✓	✓	Monoscopic	LGPL (partly GPL)	Library (libavcodec) + Binary
gstreamer	✓	✓	✓		LGPL (partly GPL)	Library
Live555		✓	✓		LGPL	Library
ccRTP		✓	✓		GPL	Library

Table 3.4: Comparison of software packages

exist that can be used for this purpose. This fits with our goal to maximize the use of standard interfaces and applications as much as possible so that the project could benefit from improvements in those applications while keeping it highly modular. These packages are compared in Table 3.4. As shown, each of these packages solves a different part of the puzzle.

These packages can be combined in several ways to create the complete toolchain. In case of libraries, the video available at user space can directly be compressed and transported. For example, libavcodec can be used to compress incoming video and the resulting video transported with help of Live555 or ccRTP libraries that support RTP streaming. Interfacing with software meant for binary distribution is not that straightforward. Software like mencoder and VLC are fairly complex, and modifying them to fetch data from the PLX driver is a daunting task. Further, linking together these software with code to retrieve the video via the PLX API is not a modular approach and does not present any advantage over using libraries. Often, they internally use libraries like libavcodec and Live555. Typically, software like FFmpeg, mencoder and VLC can encode and stream files or directly stream from a webcam. We came up with a unique approach to make video available via the Video4Linux driver interface. Thus, the cameras appear as webcams to any user space application. The several ways these packages can be combined is presented below:

1. Video4Linux (video source) + VLC / FFmpeg (compression, server) + mplayer (client, decompression, display)
2. libavcodec (compression) + Live555 / ccRTP (server) + mplayer (client, decompression, display)
3. gstreamer (compression, server) + mplayer (client, decompression, display)

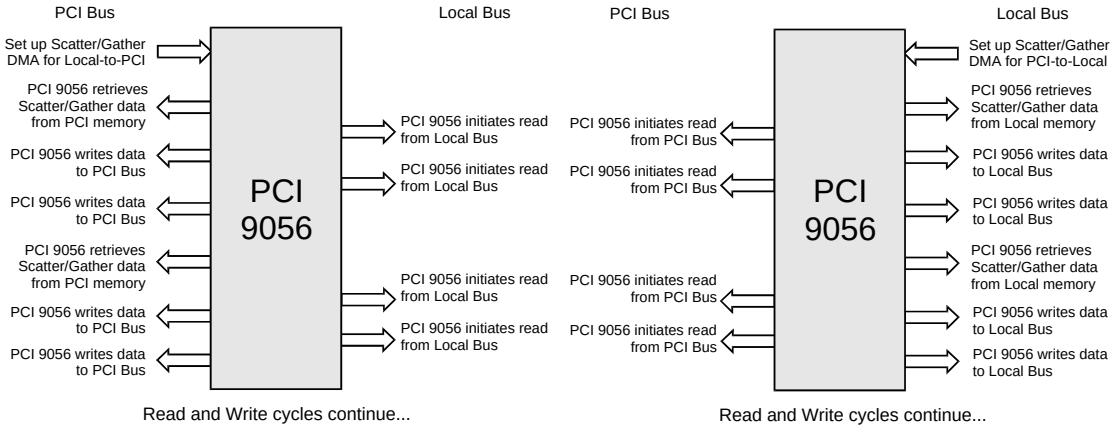
Playback of stereoscopic video is not inherently present in mplayer but a patch is available to add this capability. However, the patch was meant for an earlier release of mplayer and was tested only under Linux. Therefore, it was considered risky to patch mplayer and developing our own implementation to output stereoscopic streams using OpenGL was considered as a fallback. However, the patch was applied successfully with a few modifications. Moreover, mplayer was also used to locally display the captured video. Amongst the above choices, the first combination was selected. This is because it allowed for the structure of the software to remain highly modular by decoupling compression and transport from the hardware and drivers. Further, it also exposed the video from the cameras via a standard interface. This allows the system to be easily usable for other applications. Moreover, it also allows for independent development and testing of each part of the toolchain. FFmpeg was to be used for compression and server as it allowed for more parameters to be easily configured. This completes the entire flow of video from the PLX 9056 chip to the display at the teleoperation computer as shown in Figure 3.4. Later sections discuss this toolchain in more detail.

3.10 Video Transport Driver

As shown in Figure 3.4, the video transport driver sits between the PLX driver and the Video4Linux driver. This driver operates in user space while the PLX and Video4Linux drivers are present in kernel space. Clearly, its main task is to act as a bridge between the PCI and video streaming subsystems. It initiates the transfers of frames from the PCI bus with the help of the API provided with the PLX driver and forwards it to the Video4Linux driver. However, transporting video is not its only task. It also initiates the set-up of the cameras and the several FPGA modules. Further, it performs error checking and recovery for PCI transfers, FPGA and cameras. Therefore, it acts as a central controlling module. This makes it the most important component of the toolchain.

The PLX 9056 chip is a generic chip designed to be used for PCI cards. It communicates with the local hardware (in our case the Virtex 5 FPGA) with a local bus with separate 32 bit address and data lines. The chip allows for single or burst mode transfers. In a burst mode transfer, a number of transfers from contiguous addresses are performed after a single bus negotiation. Obviously, this significantly improves the throughput. The bursts can either be of a fixed size (4 locations) or continuous. We used continuous bursts for our system. Further, the can operate in Master, Slave or DMA modes. In the master mode, the FPGA becomes the bus master and instructs the chip to transfer data from given memory area on the FPGA to an address or a set of contiguous addresses at the PCI bus. The slave mode is the exact opposite with the master being on the other side of the PCI bus. As expected, in these modes, the master is occupied for the duration of the transfer. Therefore, we used the DMA mode in which the master places a description of the entire transfer in the registers of the PLX chip and instructs it to start a transfer. An interrupt is raised when the transfer is finished.

The DMA transfers in turn can be performed in two modes. In the DMA Block



(a) DMA transfer from Local to PCI bus (b) DMA transfer from PCI to Local bus

Figure 3.5: Sequence of events for DMA transfers with Scatter-Gather lists.

mode, the master sets the PCI and local starting address, byte count and direction and instructs the PLX 9056 chip to initiate the transfer. An interrupt is raised when the transfer is complete. In the DMA Scatter/Gather mode, the master sets up blocks of descriptors in the PCI or local memory. These blocks consists of PCI and local addresses, transfer size, direction and location of the next descriptor block. The master then enables the DMA Scatter/Gather mode (by setting a control bit in a register in PLX 9056), writes the address of the first descriptor block and initiates the transfer by setting another control bit. Figure 3.5 shows the chain of events that take place for a DMA transfer involving the use of Scatter-Gather lists.

The drives provided by PLX allows for transfers to be made either into a contiguous region of memory allocated by the driver itself, or into a user space virtual memory. Virtual memory is a memory management technique where the actual physical memory (in the form of RAM, Disks etc.) is abstracted from programs. The programs are given what they perceive to be contiguous blocks of memory as working area and the actual physical fragmentation of memory is abstracted from the programs. Further, the size of a physically contiguous block of memory called a page, is fixed (4kB for x86) depending on the architecture of the machine. Depending on fragmentation of memory, it may be possible to allocate more than one page of physically contiguous memory. For example, the kalloc function used by Linux kernel modules to dynamically allocate memory allows up to 128kB (i.e. 32 pages) of physically contiguous memory to be allocated. This process however may fail depending on the current memory fragmentation. Some kernel modules attempt to allocate memory early on (as soon as they are loaded) to avoid fragmentation but there is no guarantee that it will actually succeed. Therefore, the PLX driver only allocates a single page (4kB) of memory for DMA transfers.

During the initial design and development of the project, individual frames from memory were transferred and stored into bitmap files. For this, we used the PLX driver's

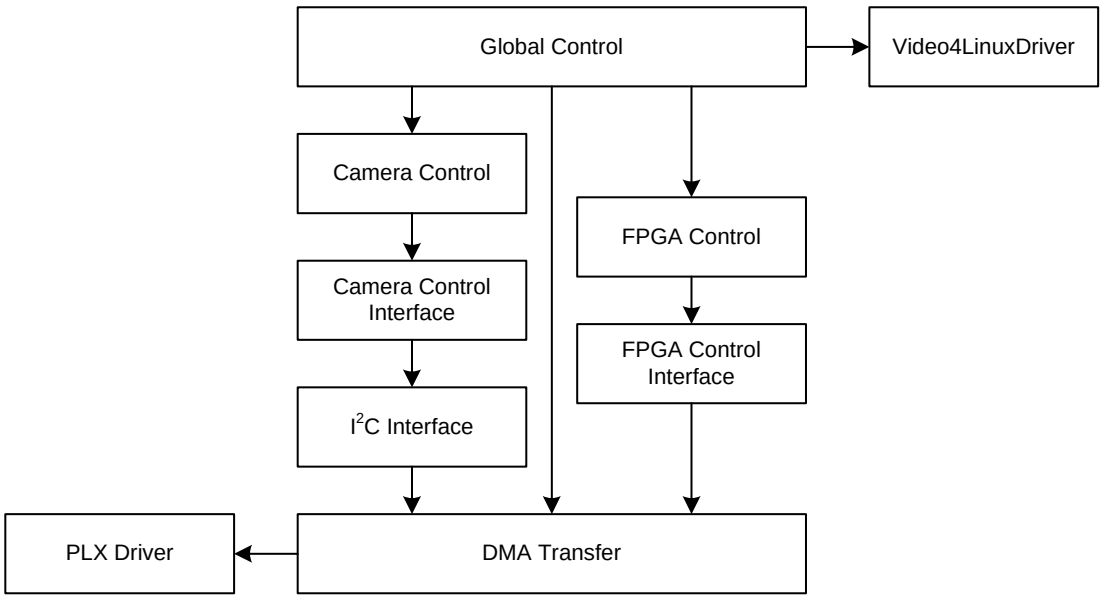


Figure 3.6: Call hierarchy of the different modules of the video transport driver

own memory for transfers using DMA block mode. For testing, the image data was replaced with incrementing numbers at the FPGA. It was soon realized that the data went missing periodically. This was caused because the time taken by the system to schedule a new transfer was longer than the time taken for the the FIFO in the FPGA to overflow. Therefore, we used memory allocated in the user space (and therefore non-contiguous) memory of the size of a single frame to transfer a complete frame in a single DMA transfer. As discussed before, in case of a FIFO overflow the FPGA holds off on storing new data and it holds the transfer of any data to the SBC until a new frame arrives. This ensures that each frame-length DMA transfer actually consists of frame data. Further, after each frame, there is window of time during which the Vsync signal is low and no new pixel data is transferred from the cameras. This blanking period allows for the set-up time of the next DMA transfer. However, transferring a complete frame at once requires the use of Scatter-Gather lists for the DMA transfer. This implies that the driver must always analyze the virtually contiguous memory, and create the Scatter-Gather list of physically contiguous pages before each transfer. As we shall see, this causes some delay before the actual transfer commences. The PLX driver was later optimized to overcome this problem.

3.10.1 Driver Architecture

Since the video transport driver interacts with all the preceding hardware components, it is important to keep it highly modular. As shown in Figure 3.6, the driver is divided into modules that interact with different hardware units. This allows for the driver to be easily modified in case one or more underlying hardware components are changed. The Camera and FPGA modules rely on the DMA transfer module for transport of their data. They are further abstracted from the DMA Transfer module with the

help of interfaces so that a change in the DMA module has minimum effect on them. Further, the I^2C interface of the cameras is dependent on the I^2C communication module in the FPGA. However, it is possible for the computer to interact with them without the intervention of the FPGA. For example USB to I^2C converters available in the market can be used for this purpose. Also, a change of cameras may mean using another bus to interact with them. Therefore, I^2C is also kept as a separate module.

Each module has a startup, reset and shut down function which is generally called by the global control function. Configuration registers in the cameras and FPGA have their corresponding data structures which contain separate elements for its internal flags and configuration bits. Corresponding interconversion functions are present to facilitate interconversion between the 8, 16 or 32 bit registers and their corresponding structures. This allows for easy configuration from a programmer's perspective which is carried out at the global control module. The driver takes simple configuration parameters at the command line at startup, translates them into module specific configuration parameters and configures the underlying modules accordingly. In addition, the driver ensures graceful exit by stopping the underlying hardware components even when killed by the user.

Error recovery is built into the driver. Apart from renegotiating the DMA channel with the PLX driver and chip, the driver also resets or restarts the preceding hardware components appropriately if a DMA transfer fails. However, with the timing optimizations and scheduling policy discussed later, recovery is not required. the current system is capable of meeting the needed transfer deadlines at all times. The only possible exception is when an interrupt takes over the CPU at the instant when a new frame transfer needs to be initiated.

3.11 DMA Transfer: Issues and Solutions

As discussed before, the video transport driver transfers frames by requesting the PLX driver to transfer data across the PCI bus for it. Lets say that the system has been transferring frames successfully for a while. Therefore, the timing of requests to transfer new frames and the incoming frames matches. At the time when a new frame transfer needs to be initiated, another process takes up the processor, it is possible that the PLX driver may no get enough CPU time to initiate the new transfer. In this case, there is a small window of time before the FIFO at the FPGA overflows and image data is lost. As mentioned before, the meaningful pixel data is followed by dummy pixels (not forwarded to the SBC) and with a blanking period where Vsync is low. Once new data starts to come in, the time (T_F) taken to fill up the FIFO is the ratio of FIFO capacity (C) and data rate (R) of incoming pixels.

$$R = \frac{\#\ \text{Pixels} * \text{Bits per Pixel}}{\text{Vsync on Time}} \tag{3.1}$$

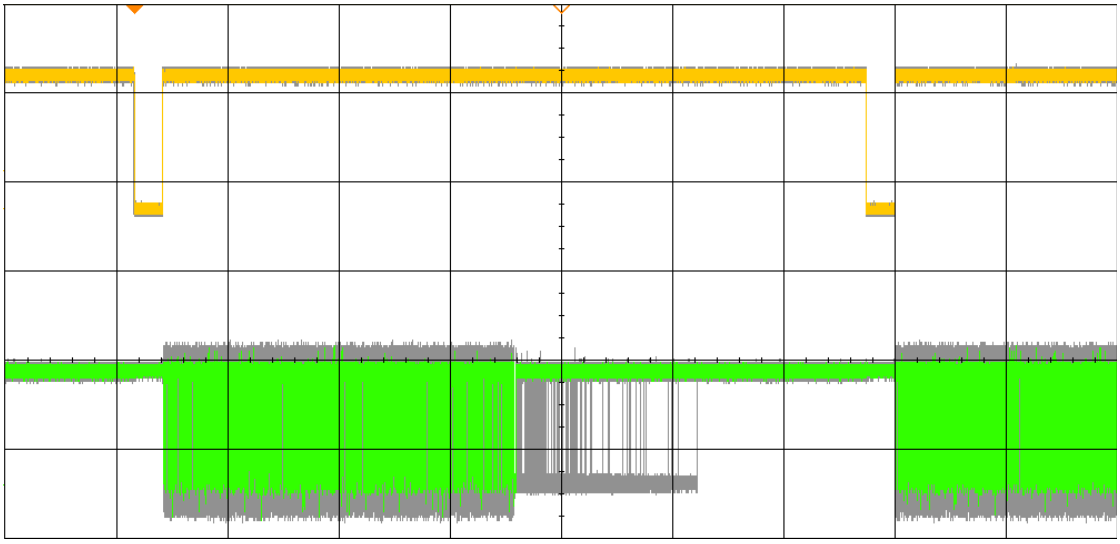


Figure 3.7: A example frame being transferred through the local bus between the FPGA and PLX 9056

$$\begin{aligned} T_F &= \frac{C}{R} = \frac{16384 * 32 * (66.69 - 3.6)}{1250 * 800 * 16} \qquad (3.2) \\ &= 2.067 \text{ ms} \\ T &= T_F + \text{Dummy Pixel Time} + \text{Vsync off Time} \qquad (3.3) \\ &= 2.067 + (66.69 - 3.6) * (800 - 384)/800 + 3.6 \\ &= 38.474 \text{ ms} \end{aligned}$$

The total time T is then the sum of T_F, time where lines with only dummy pixels are transferred and the blanking period where Vsync is low. Note that the number of pixels in the equations above is the number of pixels output by the FPGA in a full resolution frame with no dummy pixels. This is because the data rate of actual pixels does not change when the resolution is decreased. Instead, the actual pixel data is followed by a time period where no data is available to be transferred. Equations 3.2 and 3.3 are calculated at a frame rate of approximately 14.90625 fps with RGB565 format and 3.6ms is the duration for which the Vsync signal is low. These values are the same as the ones used to record the waveforms in Figure 3.7. In this figure, the orange (top) waveform corresponds to the Vsync signal and the green (bottom) waveform shows activity on the local bus between the FPGA and the PLX 9056. More specifically, the green waveform corresponds to the signal used by PLX 9056 to 'hold' of the bus. The area with variation in this signal is caused by burst transfers. After transferring the pixel data, the video transport driver requests for a new transfer and PLX 9056 holds the bus until another 4kB of pixel data is transferred. The gray area corresponds to variation in the hold signal in the past (recorded by using signal persist at the CRO). As shown, the time taken to request a new frame varies a lot.

At the resolution of 2048x768 where dummy pixels (800 – 768 = 32 lines) are

transferred for just 2.52 ms, this time reduces to just 8.19 ms. Please note that the deadlines are not affected much by single view vs. side-by-side format as there are separate FIFO's for each camera. Therefore, we need to take care of this variation in the time taken to start the transfer of a new frame and optimize the code to reduce the time taken to perform cleanup at the end of a transfer and the time taken to start a new transfer. As mentioned before, the PLX driver needs to analyze the user space memory area passed by the video transport driver to calculate the scatter gather list. This analysis primarily consists of finding out the actual physical addresses where the allocated memory is fragmented. With the help of debug messages from the PLX driver which are recorded along with timestamps in the kernel log, we found that the average time taken by the PLX driver to compute the SGL was nearly 0.54 ms. As explained later, the actual memory location wheres the frame transfer takes place is allocated by the Video4Linux driver in kernel space and the video transport driver simply maps this memory into the user space. This is one of the steps we took to remove the delay caused by copying the frame from one place to another. The Video4Linux driver can allocate several such buffers to transfer frames. However, once these buffers are allocated, their memory locations never change.

Even more importantly, the PLX driver dynamically allocates memory to store the SGL. Therefore, memory allocation failure can cause the driver to stop functioning. Therefore, we can pre-calculate the Scatter-Gather list (SGL) and look up against the starting address of the requested destination against an array of pre-calculated lists. As we shall see, the number of frame buffers is very small so the search was a simple lookup. Functionality to calculate and store an SGL was added to the PLX driver and exposed via adding to the PLX API. The video transport driver requests recalculation of SGLs at startup after it has mapped the buffers from the Video4Linux driver. As a result, the average SGL computation time was reduced to 0.17 ms.

Another optimization was to speed up the notification of transfer completion to the video transport driver. The PLX API allows the programs requesting a DMA transfer to register for a notification after requesting a transfer. Requesting notification essentially puts the requesting program on sleep until the transfer is complete. This way, the program can start a transfer, work on something else for a while and when its done, it can register for a notification and be woken up when the transfer is complete. Of course, if the transfer is already complete, it will be woken up (i.e. put in the list of runnable processes) immediately. In our case, the video transport driver has nothing to do while the transfer is being made. This can change, for example if some computer vision algorithm needs to be run on the received frame. The problem was, that upon receiving an interrupt from PLX 9056 that the DMA transfer is complete, the PLX driver did some cleanup activities and deferred the notification process in the form of a work-queue. Work-queues are a feature available for kernel modules for deferred work. This delayed the notification until the time the module got a CPU slice under normal operation. By moving the notification into the interrupt service routine, quick notification to the video transport driver was ensured.

Another optimization made was the addition of recursive transfers to the PLX driver. As mentioned before, each element of the SGL points to the next one, the last element usually carries an indication that its indeed the last element. In this case, we modify the last element of the list to point to the first one. This constitutes a never-ending transfer which is only stopped with an abort instruction. This functionality is also exposed via additions to the API. and the video transport driver also carries support to initiate such a transfer.

Such a transfer mechanism poses a data integrity problem. Say, the PLX 9056 chip is transferring to a memory location of the size of a single frame recursively. At the same time FFmpeg is also reading the same location to be compressed. In this case, the viewer will observe a tearing effect caused by the lower part of the frame being older than the upper part. This is the same as the tearing effect discussed earlier. This can be very disturbing to the user. Therefore, it must be ensured that the writer writes to a different memory location than the reader. To solve this, we can allocate a memory location whose size is a multiple of the frame size. The problem is that the software has no knowledge about when PLX 9056 has finished writing at a specific frame. The only straightforward way to ensure exclusion of the reading and writing frame memories is to include some kind of marker in the frame to indicate a frame number. The video4linux driver can then act as an arbiter by checking the frame memories each time a read request comes and point the reader to the appropriate memory. However, this kind of arbitration in the video4linux driver is left as future work.

The jitter in the time taken to schedule a new DMA transfer is caused by the Linux scheduler. Once the PLX driver receives an interrupt indicating that the transfer is complete, the video transport driver is put back in the list of runnable processes. Now its up to the scheduler to give it a slice of CPU time. However, if there are higher or same priority processes already running, the driver will have to wait in queue to get some CPU time. The problem was alleviated to some degree by elevating the static priority of the video transport driver to highest which is a nice value of -20 (-20 to 19) and a priority of 100 (100 to 139). Note that lower numbers mean higher priority. However, the scheduler also dynamically changes process priorities. Interactive processes are favored over batch processes. Interactive processes are essentially the processes that spend longer time sleeping while waiting for the kernel to perform some tasks for them. For example, waiting for an input from the keyboard. Clearly, the video transport driver is also a highly interactive process. However, while playing the video locally, mplayer also spends a long time asking the Video4Linux kernel module for new frames or calling other kernel modules to display the current frame. In fact, mplayer actually requests the next frame in a tight loop irrespective of the frame rate specified by the video4linux driver. From the kernel's perspective, both these processes are highly interactive and their dynamic priorities are increased, causing the video transport driver to sometimes miss its deadline to request a new frame, especially at the resolution of 2048x768, even when mplayer is started at lowest priority. Please note that scheduling criteria are discussed in more detail in Appendix A.

Therefore, simply running the video4linux driver at high priority was not enough to solve the scheduling problem. Fortunately, the Linux scheduler also supports running processes with different scheduling policies. Processes can be run as normal processes (discussed above), using the round-robin scheduling policy or with FIFO scheduling. Round-robin and FIFO scheduling processes have priorities ranging from 1 (highest priority) to 99 (lowest priority). Notice that this range is higher than normal processes. FIFO and round-robin processes are considered real-time processes and their CPU slices are never expired (always active, see Appendix A), irrespective of the CPU time used by them. Scheduling the video transport driver using the FIFO scheduling policy (using the chrt command) at the highest priority, it was ensured that the driver got CPU time until it willingly relinquishes the CPU or calls a kernel module. This is with the exception of interrupts in which case the driver may lose one frame but will recover in time for the next one.

The system was stress tested to ensure stability using the stress package to occupy the CPU and also under normal circumstances by manually launching several programs at once. The system is capable of transferring concatenated video at a resolution of 1024x384 and single video at a resolution of 1024x768. Concatenated video at the resolution of 2048x768 could not be used because the interconnect between the cameras and FPGA could not support the required data rate. From the perspective of DMA transfer, the PCI bus is capable of sustaining the data rates required for transporting concatenated video at the resolution of 2048x768. As explained before, concatenating the two videos does not affect the time window within which a new transfer needs to be started. A comparative study of time taken by the system to reschedule the next transfer is presented later.

3.12 Video4Linux Driver

Video4Linux is an interface used for video capture devices. It exposes a very rich set of functionalities to the user applications that source the video from its API. It is not meant as a method to actually retrieve the video from the hardware, rather it is only meant to create a common platform from which any user space application can obtain the retrieved video. It also presents a standard interface for applications to configure the device such as changing channels for a TV-tuner, negotiate the video format, colour space etc. It has become a de-facto standard for video capture devices and is used by most user space programs to obtain the captured video. However, little documentation is available for developing drivers with it. The author developed an interfacing driver with the Video4Linux2 API.

As mentioned before, in order to avoid tearing, frames must be buffered such that the reader and writer work with separate frames. Video4Linux2 provides a method for user space applications to request and return frame buffers shown in Figure 3.8. Initially, applications retrieve information about several capabilities of the device and the driver such as resolution, colour format, frame rate and number of available frame buffers. If several options are available, such as multiple colour

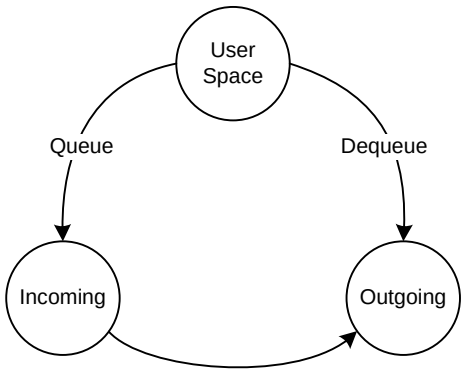


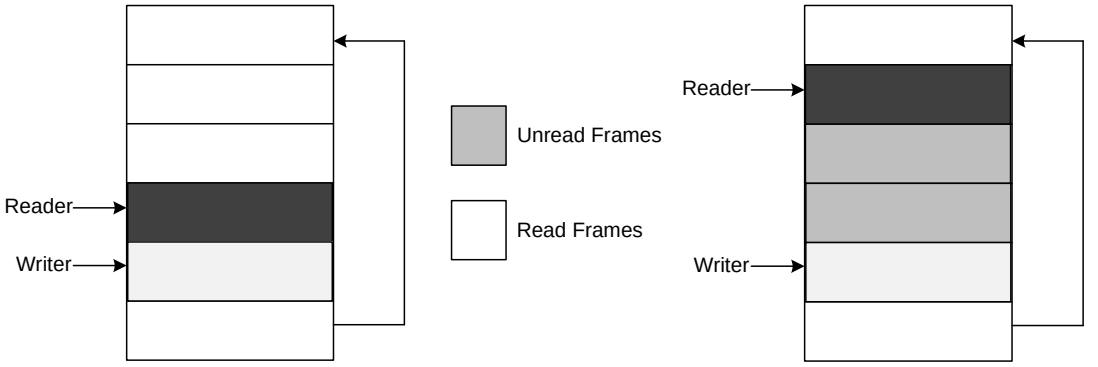
Figure 3.8: Video4Linux frame transaction with user space applications

formats, the application may request one of them. Once all negotiations are complete, applications can request to use a frame buffer (dequeue), processes it and return it back (queue) to the driver. While a frame is queued, the driver is free to write a new frame at that location. The API only requires that a driver implement these functions, the actual management mechanism for frames is left up to the developer.

Several schemes were considered for managing the frames. It was chosen to use a Circular Queue for management of frames. In this case, the driver acts as an arbiter between the video transport driver and compression and display applications. It maintains variables pointing to the position of the writer (video transport driver) and reader (FFmpeg or mplayer). The queue management scheme is described below:

- Initially, the writer position is initialized as one frame ahead of the reader.
- When the writer/reader requests a new frame buffer, it is given the next location in the queue unless the next location is occupied by the reader/writer.
- If the next location is occupied, the writer/reader is given the same location.

Although this scheme is inherently simple, it adheres well to our goal of minimizing latency while keeping the underlying computations very lightweight. The reader is always much faster than the writer. A new frame is written only when one is available from the hardware. Mplayer attempts to output a new frame as fast as possible. In case of FFmpeg, it proceeds at a rate of 100fps which is specified by us. However, it only compresses and streams the frames at a rate of 22fps. Timing measurements of this FIFO showed that on an average, the time taken between the video transport driver requesting the next write position and FFmpeg requesting a new buffer is approximately 97μS. Therefore, a negligible delay is induced at this step. The number of buffers does not affect the system. The only exception to this case would be when the queue is full, in which case the writer continues to write at the same location again. In this case, when the reader does catch up, the user would see a jerk in the video as some time would elapse between the one but latest and the latest frames. Therefore, the queue is usually kept at a size of 5 frame buffers. Figure 3.9 shows the circular queue with a normal case where the reader is immediately behind the writer and the



(a) Reader is just behind the writer (b) Reader is lagging behind the writer

Figure 3.9: Circular queue used for managing frame buffers

case where the reader is lagging.

Further, latency was also minimized with the help of memory mapping. Drivers can provide data to applications by means of two methods. In the first case, the writer writes to the driver as if it were writing a file. This means that the video transport driver must first cache a frame locally, and then write it to the Video4Linux driver where it is stored again. Then the reader reads this frame into its own local memory with a read call. Clearly, this will significantly increase latency as the frame is copied at each step. Instead, we used memory mapping to transfer frames. The writer and reader map the memory allocated by the video4linux driver into user space. DMA transfers are performed directly into the buffers and the reader can use the frame directly. The frame buffers themselves are allocated at initialization time to avoid any problems later because of memory allocation failure. Further, as the queue-dequeue mechanism was unnecessary for the writer, we implemented a custom communication mechanism which only requested the next location to write.

3.13 FFmpeg and mplayer

As discussed before, FFmpeg was used for compression and streaming of the stereoscopic video. It is one of the leading open source projects for media applications and is its library for compression and decompression is widely used in other projects. We configured FFmpeg to use video4linux as input. Although applications using the video4linux API are expected to negotiate the frame rate with the driver, it was found that this is hardly the case. Configuring FFmpeg to source the incoming video at a high frame rate of 100fps ensures that the latency is minimized. An asynchronous method to transport frames with video4linux as an arbiter is not theoretically the ideal solution but it ensures a high degree of modularity. Therefore, it was chosen above other, more tightly coupled approaches. Further, it was configured to compress and steam the stereocopic video using the RTP protocol. MPEG-4 compression with intra frames only was used. As discussed earlier, this minimized the processing requirement and reduced

latency caused by the need to buffer multiple frames. FFMPEG also allows control over the level of compression by specifying the quantization parameter. Alternatively, it can also configured to output the same quality video as input. It was found that the processing requirement for compression was quite high and FFMPEG consumed most of the available CPU time. To ensure smooth operation of the more time critical DMA operation, we ran FFMPEG at the lowest priority.

On the teleoperation machine, mplayer was used to display the video. As mentioned before, we used OpenGL output with quad buffers to output separate left and right views at the graphics card. The decoded frame is first split into its constituent left and right views. Next, these are written into appropriate buffers followed by an instruction to draw these buffer. This effectively swaps the two double buffers, allowing the graphics card to display the contents of the front buffers while the back buffers are again written with the contents of a new frame. In order to minimize latency, we disabled caching of frames at mplayer.

4 Case Study

"A subtle thought that is in error may yet give rise to fruitful inquiry that can establish truths of great value." – Isaac Asimov

In the previous chapter, the toolchain for a stereoscopic remote vision system was described in detail. In this chapter, a case study comparing the timing and delay effect of the several aspects of the toolchain is presented. Further, we present a study of user performance in a teleoperation-like scenario while varying several parameters.

4.1 DMA Transfer Time

After configuring the FPGA and cameras, the video transport driver spins in a tight loop performing the following tasks:

1. Request a new frame buffer from the Video4Linux driver. This is the memory location where the frame will be transferred.
2. Request the PLX 9056 chip to transfer a new frame (via the PLX Driver). This involves the PLX driver computing the Scatter-Gather list first. Once its done, the first address in the SGL is sent to PLX 9056 and the transfer begins.
3. Request the PLX Driver to notify it when the transfer is complete. This task does not influence the timely behavior of the system as the transfer proceeds independent of it.
4. Receives a notification that the DMA transfer is complete.

This process is illustrated in Figure 4.1. Please note that the figure does not include step 3 shown above because it does not influence the timely behavior of the system. The time between the end of the transfer of a frame and the beginning of the next transfer

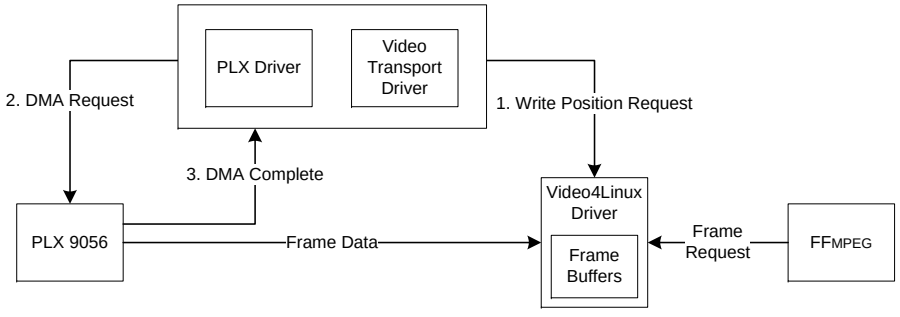


Figure 4.1: Sequence of events involved in DMA transfer of a frame

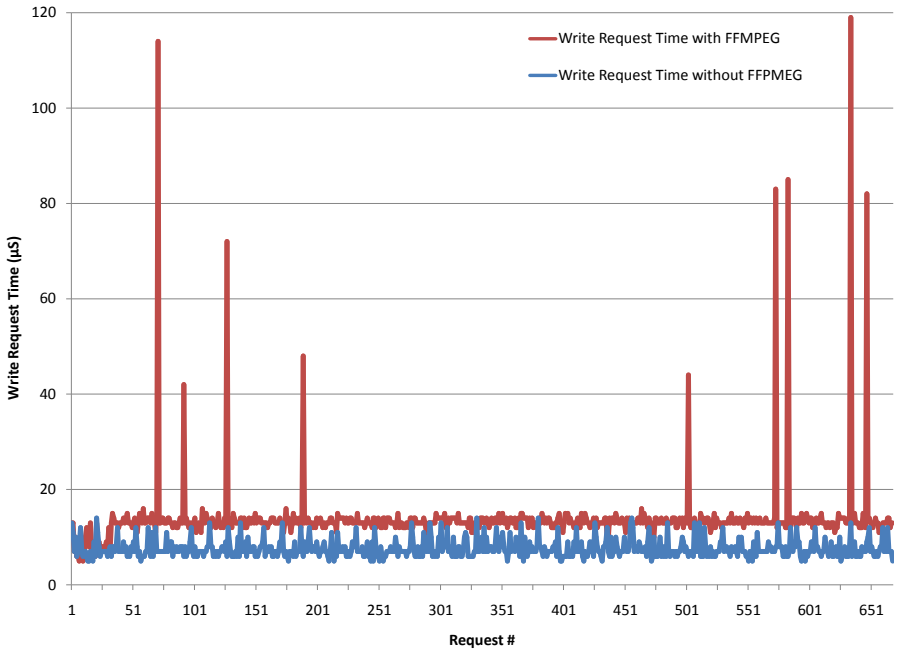


Figure 4.2: Comparison of time taken for Video4Linux Driver to give the next frame buffer with and without FFmpeg

decides whether the FIFO overflows and therefore a whole frame is lost. Therefore, the time taken for these tasks is compared here for different scenarios.

Figure 4.2 compares time taken by the Video4Linux driver to give the next frame buffer to be written. These measurements are taken at the video transport driver. A timestamp was recorded before and after the execution of this function. Adding FFmpeg to the equation increases the average time taken for this function from 7.59μS to 14.09μS. Please note that these values are so small that they may be affected by variations in the execution time of the function used for generating timestamps (gettimeofday). Nevertheless, the plots do show some trends. The high peaks at the beginning and end of the plot are caused by the start and end of FFmpeg which requires the Video4Linux driver to do some initialization and cleanup oriented tasks. A call from the video transport driver should preempt the call from FFmpeg, but the context switch time is still involved. Further, some parts of the Video4Linux driver involve holding spinlocks for a short duration. Holding spinlocks disables kernel preemption. Spinlocks are lightweight semaphore-like constructs available to kernel modules to preserve atomicity of its operations where required. The spinlock and context switch times may cause such peaks.

Execution time measurements taken at the Video4Linux driver are consistent with the above explanation. When the function to give the next write position has started, it means that any spinlocks being held have already been released; and the context switch has also taken place. So the time spent inside this function is expected to remain nearly

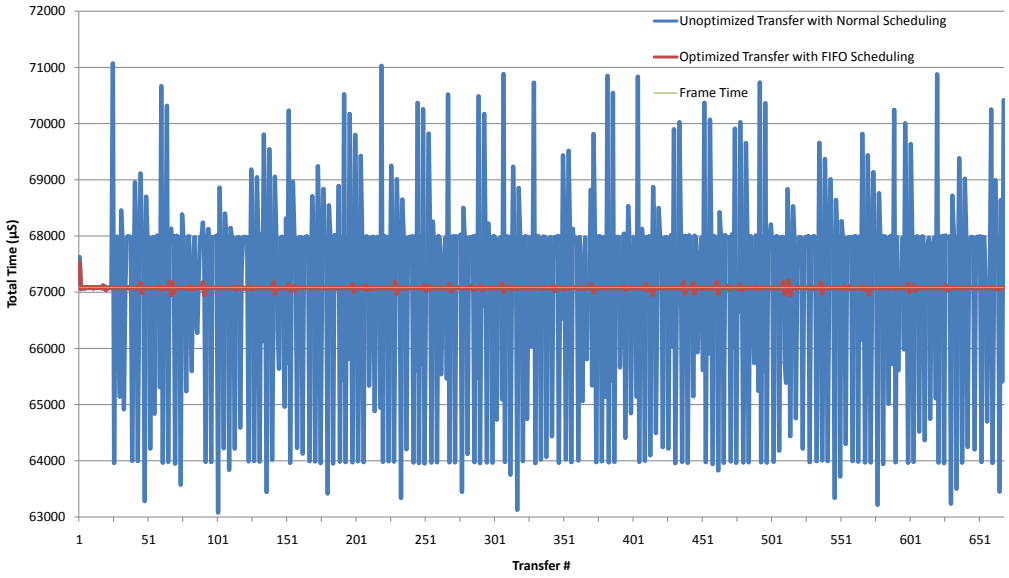


Figure 4.3: Comparison of optimized transfer with FIFO scheduling and unoptimized transfer with normal scheduling at highest priority

constant. This was indeed the case, and the average execution time was found to be 3.2μS irrespective of whether FFmpeg was running or not.

As discussed in the earlier chapter, time taken by the PLX driver to compute the Scatter-Gather list was also measured. For the unoptimized driver, the average time taken was 0.54ms and that for the optimized driver was 0.17ms. These values were measured at the resolution of 1024x384. This time increases with frame size and was quadruple (2.16ms) for the resolution of 2048x768. For the optimized driver, this time remained constant as there is no additional load with a larger SGL. Please note that this is the time spent inside the function that calculates the SGL, other tasks are also performed at the beginning of a DMA transfer and the average total time between receiving a request for DMA transfer from the video transport driver and actually initiating the transfer was found to be 0.61ms (2.86ms for 2048x768) for the unoptimized and 0.25ms for the optimized driver. Once again, this the actual time taken by the driver, and does not incorporate the additional delay caused by scheduling, context switches and spinlocks.

Figure 4.3 shows the comparison of unoptimized DMA transfer with high priority with normal scheduling policy and optimized transfer with FIFO scheduling policy. Both the plots were recorded with FFmpeg running. Further this is the total time for one frame i.e. it incorporates all the steps mentioned above. The effect of switching to the FIFO scheduling scheme is clearly visible here and the variance is reduced from 3207372μS^2 to 851.52μS^2. The initial period of low values is for the time period when FFmpeg was not started yet and shows how even normal scheduling policy with unoptimized driver is sufficient at low CPU load.

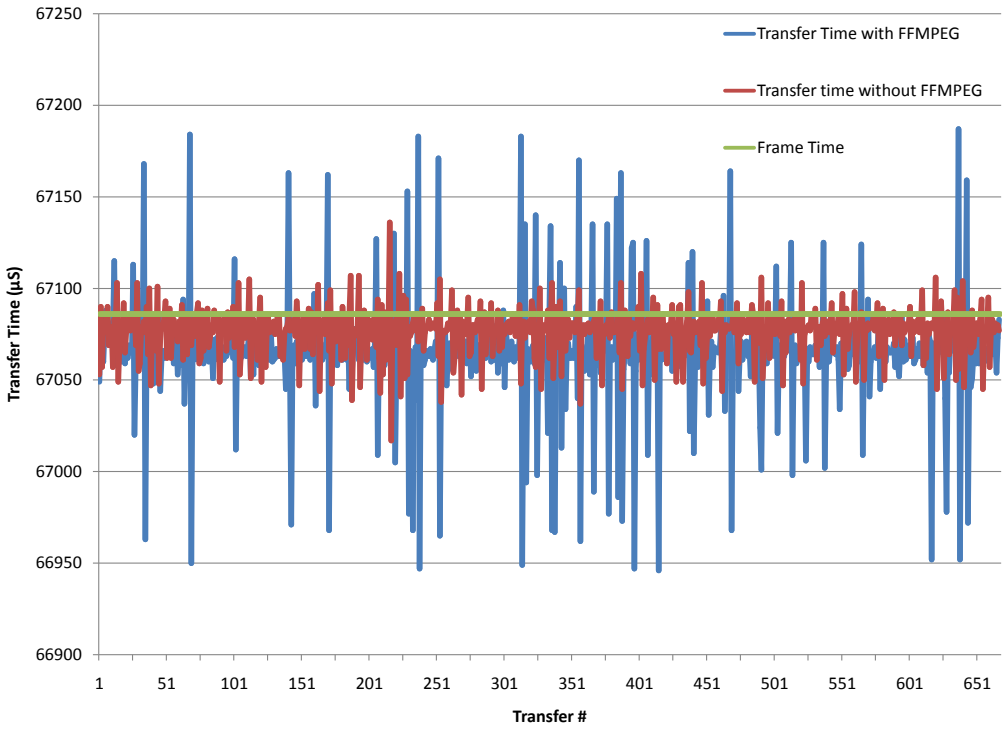


Figure 4.4: DMA transfer time with and without FFmpeg

Figure 4.4 shows the time taken to transfer a frame with and without FFmpeg running. The transfer time referred to here is the time starting at step 2 and ending with step 4 in the above list of tasks. These measurements were taken with a frame rate of 14.90625 frames per second at a resolution of 1024x384(concatenated views) with the optimized driver running with FIFO scheduling policy. Unless mentioned otherwise, the conditions for further measurements are the same. The measurements were made with the help of timestamps (gettimeofday function) taken just before step 2 and at the end of step 4 in the list of tasks above.

Note that the average case transfer time with FFmpeg is much lower than that without it. This is because the measurements are taken from the time a transfer request is initiated and the time it completes. Consequently, the time taken waiting for the new frame data to arrive is also incorporated here. Further, transfer taking a long time sometimes delays the next transfer as well which shows up as taking less time than normal. This does not affect the quality of our measurements as variation in the time is what matters and the worst cases can cause FIFO overflow. The variance of transfer times without FFmpeg was $493.6\mu S^2$ and that with FFmpeg was $1230.7\mu S^2$.

Note that it is difficult to deduce how late the actual transfer was from these graphs. Longer transfer time may mean that the transfer was rescheduled faster than the average case. Therefore, it spent longer time waiting for the data to arrive.

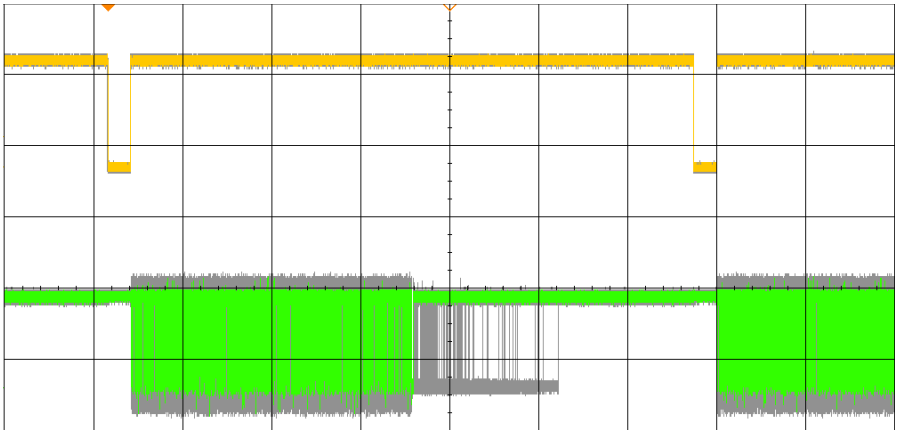


Figure 4.5: Frame transfer at the local bus interface at the resolution of 1024x384 (non-optimized and normal scheduling, 10mS per division)

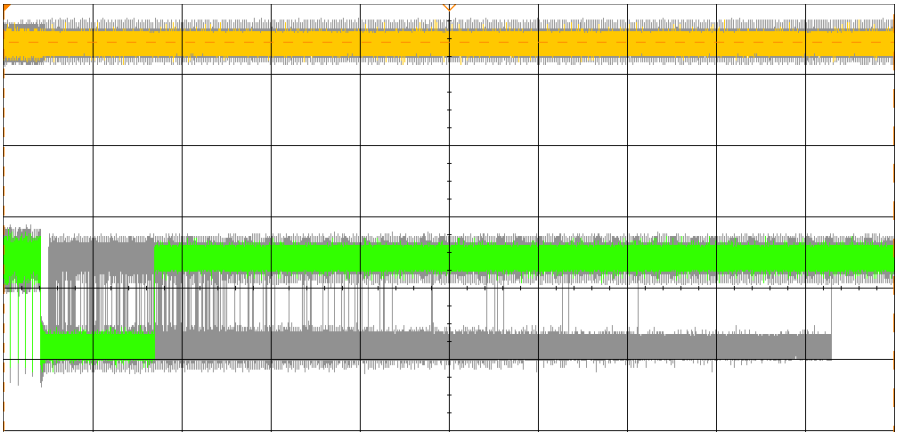


Figure 4.6: Detailed Frame transfer at the local bus interface at the resolution of 1024x384 (non-optimized and normal scheduling, 2mS per division)

Or it could mean that the transfer was completed but spinlocks and context switch elongated the time taken for the video transport driver to be notified that the transfer is complete. The opposite applies to shorter transfer times. Therefore, it is not possible to tell how long the actual transfer took, or how late the transfer was. The transfer was analyzed from the perspective of the FPGA to find more meaningful information.

Figure 4.5 shows the transfer of a complete frame for the non-optimized driver with normal scheduling scheme. The orange waveform shows the Vsync signal. This signal is high when a frame is being transmitted and low for 3.6ms in between frames where no data is transmitted. Recall that dummy pixels are transmitted when the cameras are configured to operate at a lower resolution. The green (bottom) waveform shows activity on the local bus between the FPGA and the PLX 9056. More specifically, the green waveform corresponds to the signal used by PLX 9056 to 'hold' of the bus. The

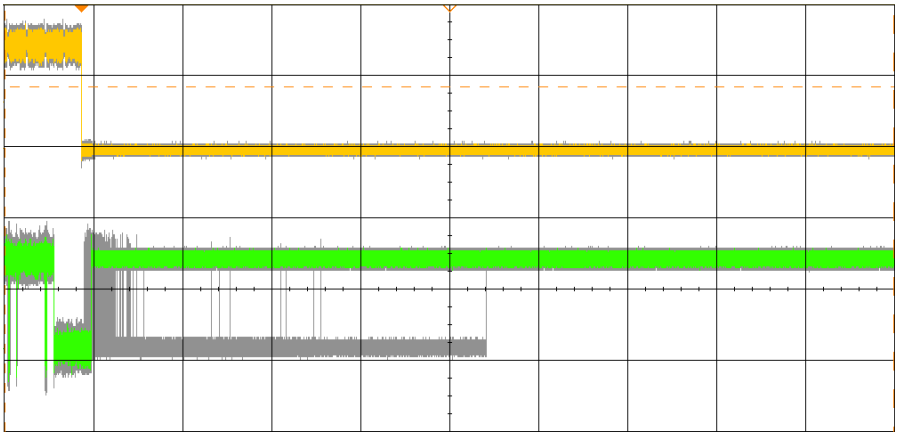


Figure 4.7: Frame transfer at the local bus interface at the resolution of 2048x768 (Optimized and FIFO scheduling, 200μS per division)

Resolution	Request Write Posn.	Init Transfer	Notify Tr. Complete	Time Window	Remaining Time
Optimized Driver:					
1024x384	3.2μS	0.25mS	0.1mS	38.74mS	38.387mS
2048x768	3.2μS	0.25mS	0.1mS	8.19mS	7.837mS
Unoptimized Driver:					
1024x384	3.2μS	0.61mS	0.34mS	38.74mS	37.787mS
2048x768	3.2μS	2.86mS	0.34mS	8.19mS	4.987mS

Table 4.1: Actual time taken for different tasks involved in transferring a frame to the frame buffers in Video4Linux Driver

area with variation in this signal is caused by burst transfers. The gray area corresponds to variation in the hold signal in the past (recorded by using signal persist at the CRO). As shown, the time taken to request a new frame varies a lot. This data was recorded at a resolution of 1024x384 with side-by-side stereoscopic format at a frame rate of 14.90625fps. Figure 4.6, shows a CRO recording taken at the same conditions at a higher time resolution of 2ms per division. In this recording the worst case time time taken to reschedule a frame transfer is approximately 16.86mS. Conversely, figure 4.7 shows a similar plot taken with video being transferred at the resolution of 2048x768. In this case, FIFO scheduling policy is used along with the optimized driver. In this case the worst case time to reschedule a transfer is just 964μS. This clearly shows how using FIFO scheduling improves the timely behavior of the system.

The time window during which a new transfer must be started three factors: frame rate, frame size and how much data is buffered at the FIFO in the FPGA. As shown in Equation 3.3, this time is 38.474mS for the resolution of 1024x384 and 8.19ms for

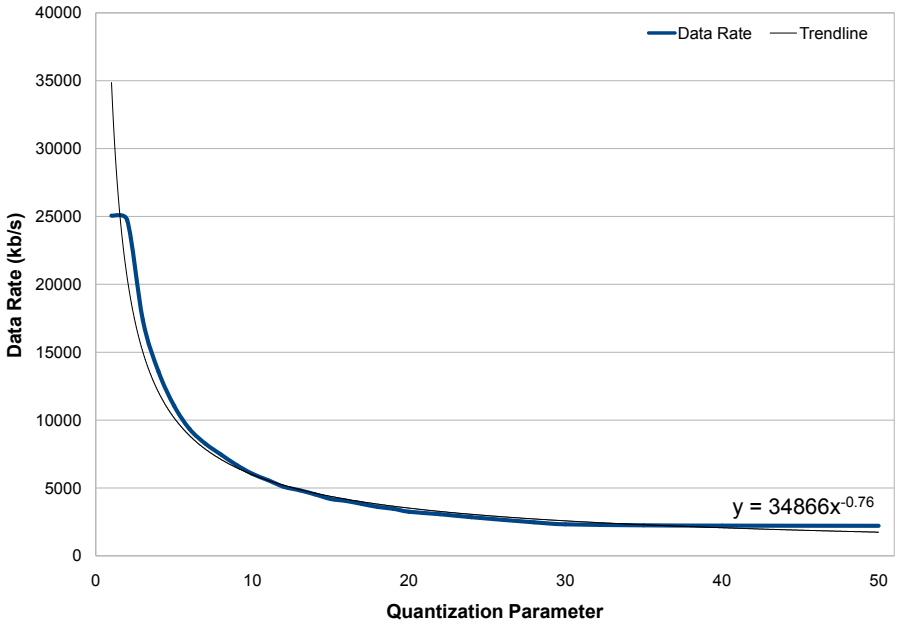


Figure 4.8: Variation of Data Rate with Quantization Parameter

2048x768. As explained earlier, this time is not affected if a concatenated or single view is transferred. This means that this window is same as above for resolutions of 512x384 and 1024x768 respectively. Based on the above measurements, the amount of time spent by the drivers performing different tasks is presented in Table 4.1. The table lists these events in the order of occurrence from left to right. The actual transfer takes place in between transfer initialization and notification that the transfer is complete. Please note that the times mentioned here are the average actual times taken performing these tasks. They do not include the time spent during scheduling, context switches and while waiting for spinlocks to be released. Therefore, some of the remaining slack time is used up because of these overheads and the total time to schedule a new transfer comes to a maximum recorded value of 0.964mS for optimized driver with FIFO scheduling and 16.84mS for normal scheduling with unoptimized driver. Therefore, maximum recorded times of 0.817mS (FIFO scheduling policy, optimized driver) and 15.887mS (normal scheduling policy, unoptimized driver) are used for these overheads.

4.2 Effect of Compression

Data rate and end-to-end delay were measured while varying the Quantization Parameter of compression at FFmpeg. Recall that Quantization Parameter typically varies from 1 to 51 for MPEG-4 compression and increases the compression ratio at the expense of output quality. Figure 4.8 shows the variation of data rate with respect to QP. As shown by the trendline (black), data rate nearly decays as a power of the Quantization Parameter. Data rate was available directly from FFmpeg but measuring delay was not so straightforward.

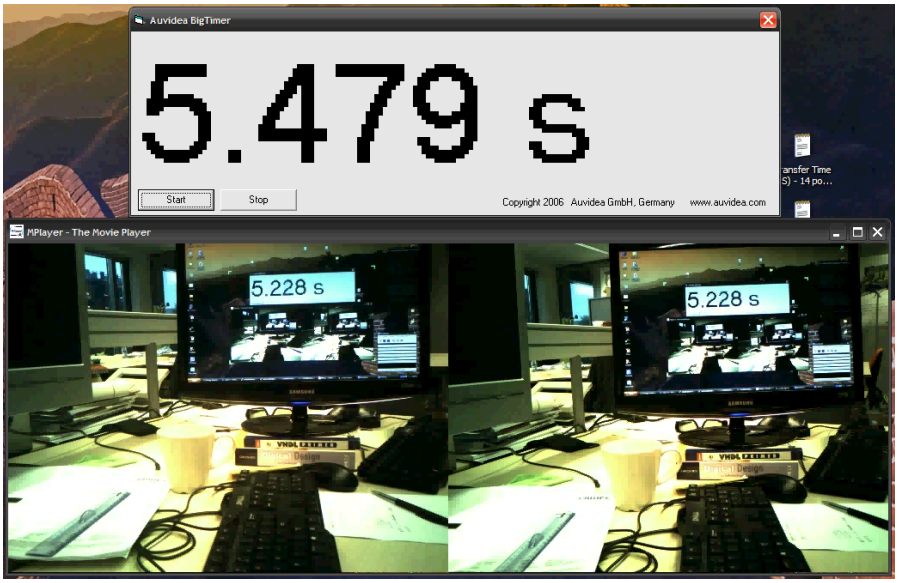


Figure 4.9: A screenshot depicting method for measuring delay

As the reader is aware by now, the complete remote vision system consists of many components much of which cannot be modified. Therefore, measuring the end-to-end delay is a daunting task. For example, it may be possible to measure delay from the point at which the viedo enters the video transport driver with the help of timestamps. On the other end, mplayer can be modified to read and compare these timestamps. Network Time Protocol (NTP) can be used to synchronize the two computer's own time. However, we still cannot account for the delay from cameras, processing done at FPGA, time taken for video to reach the screen etc. Further, this would involve modifying several software packages to take these measurements. Therefore, a better approach is to measure the delay of the system as a black box. This was done by pointing the cameras to the screen at the teleoperation machine. The screen shows a timer with millisecond accuracy. A video of the screen as seen by the cameras is streamed to the same screen with the remote vision system. Finally, the contents of the complete screen are recorded into a file using software. The difference between the actual timer and the timer as seen by cameras which shows up in the streamed video then yields a round trip delay of the system. A image showing the actual timer along with the video of the timer as seen by the cameras is shown in Figure 4.9.

The video shown in this figure is in its original side-by-side form for clarity. The actual measurements were taken with the video being shown stereoscopically. One precaution taken was to measure the delay only at times when a new frame arrives (change in the timer value is observed in the video). This was done because a frame persists at-least for a time of of 1/frame-rate which is approximately 67.08ms. If a reading is taken at a random instant, anything from 0 to 67.08ms may be added to the actual measurement. Since the refresh rate of the screen is 60Hz, the accuracy of

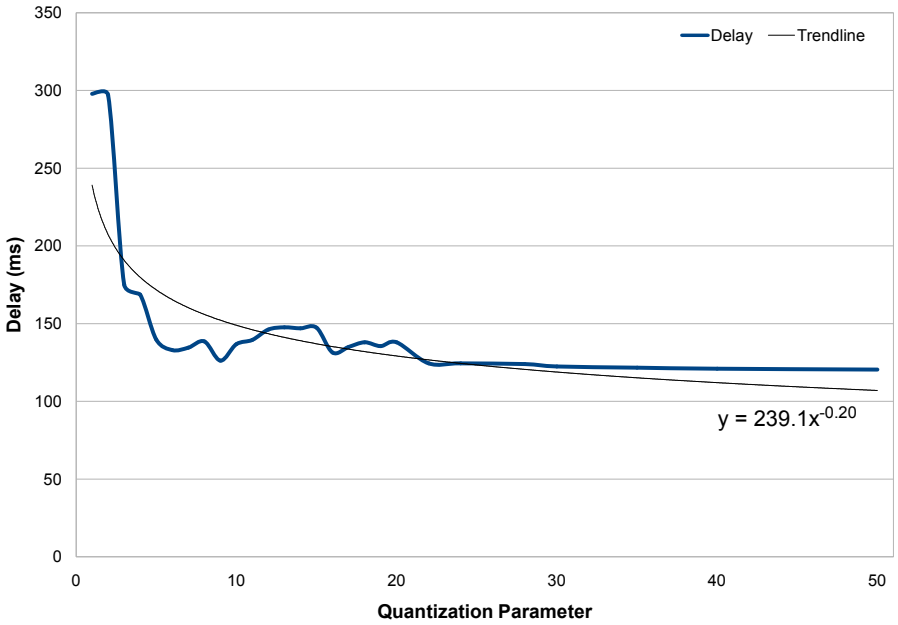


Figure 4.10: Variation of Delay with Quantization Parameter

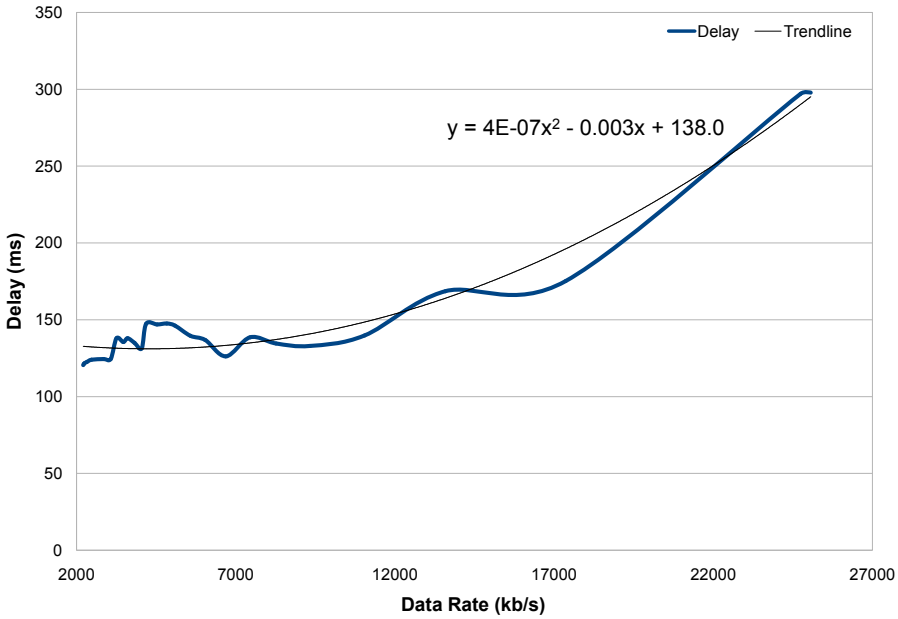


Figure 4.11: Variation of Delay with Data Rate

these readings may also suffer from the time period taken to refresh the screen 1/60 = 16.66ms. Therefore, 16.66ms is our margin of error.

In order for these readings to be statistically accurate and independent of any jitter in delay, an average of 10 measurements was taken for each QP. However, the individual

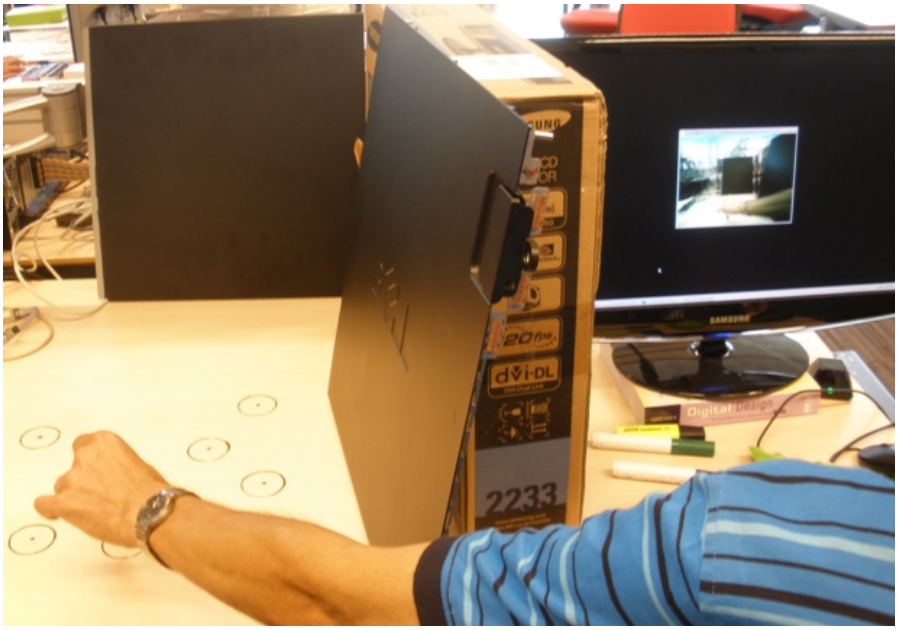


Figure 4.12: Photograph of a user test

readings were not found to vary much. This kind of measurements may suffer from discrete behavior of the system, such as the refresh rate of the screen and untimely behavior of software may cause inaccurate measurements. However, the main goal is to compare the effect of compression on delay and not to establish highly accurate delay measurements. This task is done well by our black-box measurements and the results are shown in Figures 4.10. As expected, delay decreases as we increase compression. This is not only because of decreased network latency but also because less processing is required as QP is increased[34]. Further, Figure 4.11 shows the variation of Delay with Data Rate. As shown by the trendline, delay of the system approximately shows a polynomial growth with the data rate.

Increasing compression has a two fold effect of decreasing delay while also decreasing the quality of the video. Therefore, it makes an interesting study to see how users perform when these factors are changed together.

4.3 User Survey

As explained earlier, several factors may affect the performance of a teleoperator. The focus for this thesis has been to maximize teleoperator performance. A survey with performance of users was taken while varying delay, depth perception and video quality. For this, the users were presented with a teleoperation-like scenario in which the users were required to perform a task while using the developed remote vision system. Several targets were marked on a table and the cameras were placed overlooking the table. Next, the users were asked to place a mark as close to the target as they could with

User #	Average User Error (cm)			
	297.8 ms	364.88 ms	431.96 ms	499.04 ms
1	0.48	0.52	0.88	3.4
2	0.67	0.88	1.78	3.16
3	0.18	0.22	0.32	0.78
4	0.36	0.44	0.36	0.44
5	0.26	0.3	0.56	0.78
6	0.21	0.34	0.36	0.9
7	0.82	0.94	1.26	1.72

Table 4.2: User Test results for various delays(ms)

the help of a magic marker while viewing the table and their own arms with the help of the stereoscopic remote vision system. The distances from the placed marks and the targets were then recorded. The delay and compression (QP) were then varied to measure the corresponding variation in user performance. The users were also asked to perform the same test while viewing the same scene with a monoscopic video feed. Figure 4.12 shows a user performing the test. The photograph shows the stereoscopic video on the teleoperation machine on the right and the targets on the left.

Before performing the actual test, the users were asked to practice the task until they were comfortable with the scenario. This made sure that the test results were not affected by changes in user's proficiency as they repeated the same task. The test was performed with 7 adult users. Further, in order to keep the results statistically accurate, every test was performed 5 times for each variation of a parameter and the resulting measurements were averaged. Delay was varied by buffering the frames and therefore in steps of 1/frame-rate (67.08ms). The based delay is derived from the value of delay for QP value of 1 from the previous section. The video resolution was 512x384 stereoscopic (i.e. 1024x384 resolution of actual video) for all cases. Quantization Parameter was kept constant at 1 throughout the experiment. Table 4.2 shows the result of this test. Note that lower values signify better accuracy. A shown in the table, the accuracy of users for the same value of delay varies a lot. User performance can vary a with a lot of factors such as their visual acuity, overall dexterity, concentration etc. However, the overall trend still shows a pattern. In general, accuracy decreases with delay. This was expected. We are more interested in the variation of accuracy with delay.

In order to derive more definitive values from the recorded user accuracies. In order to find a norm amongst all the users, the minimum (most accurate) was subtracted from each value. Followed by division from the range of the user's error (maximum - minimum). This normalized each user's average error as values ranging from 0 to 1. Therefore:

$$\text{Normalized Error} = \frac{\text{Error} - \text{Minimum Error}}{\text{Maximum Error} - \text{Minimum Error}}$$

This normalized error was then averaged across all users for every delay. The resulting

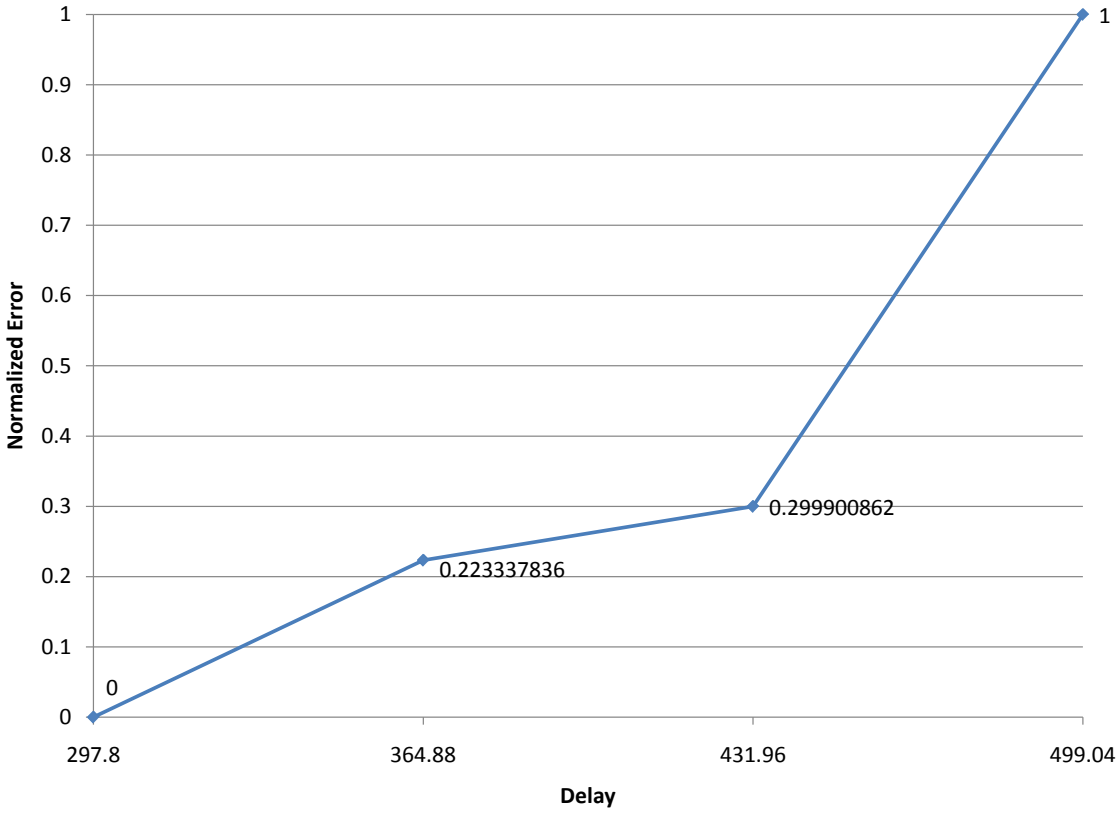


Figure 4.13: Variation of Normalized user error with Delay

User #	Average User Error (cm)					
	1	2	4	8	16	32
1	0.48	0.5	0.3	1.1	1.06	1.66
2	0.67	0.68	0.45	1.04	1.35	1.6
3	0.18	0.21	0.15	0.35	0.46	1.83
4	0.36	0.1	0.26	0.3	0.44	0.74
5	0.26	0.32	0.22	0.18	0.2	0.8
6	0.21	0.18	0.13	0.17	0.23	0.94
7	0.82	0.78	0.64	1.34	1.73	2.1

Table 4.3: User Test results for various levels of Quantization Parameter (compression)

variation is plotted in Figure 4.13. As the figure shows, accuracy does not scale linearly with delay and decreases more rapidly for higher values of delay.

Another parameter for user accuracy is the quality of video. The level of compression (varied with the Quantization Parameter) is an important factor affecting video quality. However, it also affects data rate and delay. Variation of delay with quantization parameter has been discussed in the previous section and delay was found

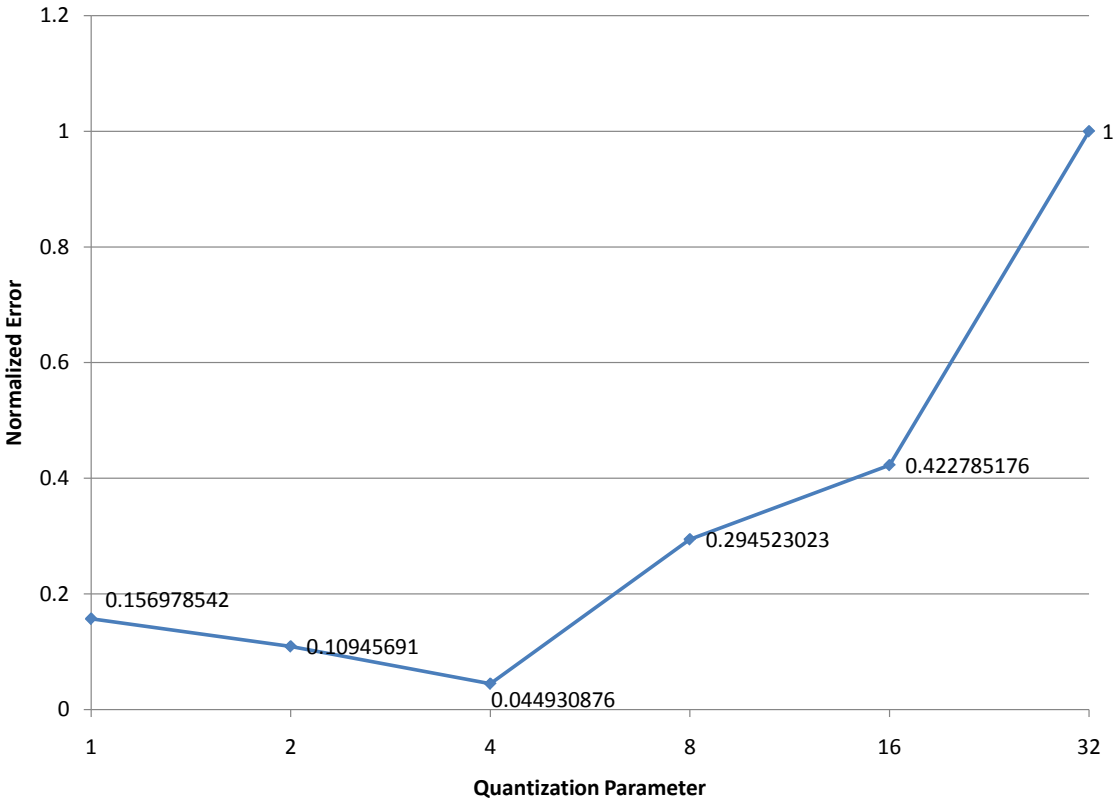


Figure 4.14: Variation of Normalized user error with Quantization Parameter (compression)

User #	Stereoscopic	Monoscopic
1	0.48	0.98
2	0.67	3.12
3	0.18	0.38
4	0.36	1.8
5	0.26	1.16
6	0.21	0.92
7	0.82	3.46

Table 4.4: Comparison of user error between monoscopic and stereoscopic video

to approximately decay as a power function of QP. However, as shown above, user performance improves with decreasing delay. The overall effect should be such that there should be an optimum value where user performance can be maximized. For this reason, compression is an important tradeoff that needs to be made to optimize user performance and was explored during this survey. During the initial testing of the system, it was found that users were much more comfortable in the lower range of QP. So the user survey was performed with QP changing as a power of 2 in order

to have better resolution of results in the lower range of QP. The results are shown in Table 4.3. The same normalization method was used for the obtained results as well. As can be seen from Figure 4.14, the optimum value for compression was found at the Quantization Parameter of 4.

Finally, users were also asked to do the same task while viewing a monoscopic video feed. Measurements were taken in the same conditions are the base case for delay that is, a delay of 297.8ms and QP of 1. The results comparing the error in placing marks for monoscopic as compared to stereoscopic case with delay of 297.8ms is shown in Table 4.4. The results were quite encouraging, the average error was 3.83 times for monoscopic as compared to stereoscopic video.

5 Conclusion and Future Work

"The true delight is in the finding out rather than in the knowing." – Isaac Asimov

5.1 Conclusion

Teleoperation is a a field of technology places high demands on the underlying system. Not only should a teleoperation system create a sense of presence at the remote location, it must also allow fluid control of actuators at the remote site. This makes delay a key factor for performance of the teleoperator.

Several methods were explored for capture, compression, transport and display of 3D video. For capture, very large arrays of cameras have been used to capture multi-view video. Compression of the resulting video streams may involve the use of technologies that stem from temporal correlation used in monoscopic video and from computer graphics. Much of these technologies involve a high amount of processing power and bandwidth which conflicts with the goals of teleoperation.

After exploring and exhaustive set of technologies pertaining to remote vision, the set of technologies was pruned according to the requirements derived from factors of human comfort for 3D video and performance in teleoperation. By doing so, this report presents a qualitative analysis of 3D remote vision technologies. The selected technologies attempt to make the best of the limited resources of an embedded system. As hardware gets smaller and more powerful, more computationally intensive methods may become feasible. This research, analysis and decision process can be re-used in the future to create better 3D remote vision systems.

The report also presents a study of delay and compression on teleoperator performance. As per the knowledge of the author, a comparative study of these parameters for such systems is not present in literature. As discussed earlier, increasing compression has the dual effect of reducing video quality and reducing delay. These factors reduce and improve teleoperator performance respectively. Therefore, tradeoff was made between video quality and delay based on the results of a user survey.

The stereoscopic remote vision that has been created can be combined with robotic arms and mounted on a wheelchair. The resulting system can be used to remotely assist disabled individuals perform daily tasks, improving their quality of life.

5.2 Future Work

The current system only utilizes two cameras. As discussed earlier, adding a third camera to the system will allow for a monoscopic field of view of 180°. The 3 video feeds can be combined the same way as the two current camera views. The resulting video would need to be transformed to remove lens distortions which can be done at the client side where ample processing power is available.

As mentioned earlier, the display system was selected while keeping extensibility in mind. Presently only one screen is used but the display system can support up to 9 screens simultaneously. As a first step, two more screens can be added on each side to create a curved display system to surround the teleoperator with video from the remote environment. The author expects that such a system would increase the sense of presence for the teleoperator. The two additional screens would show monoscopic video from the third camera while the center screen shows stereoscopic video. Such a system would enable the operator to use depth perception to perform tasks while being aware of the surroundings.

Head tracking can be used to capture the orientation of the operator's head. This information can be sent to the remote site and used to turn the complete camera setup towards the orientation intended by the operator.

As discussed before, the current system requires that each frame must be individually requested for transfer over the PCI bus using DMA. Support for continuous transfers was also added to the drivers. This makes the PCI chip continue to transfer data over the PCI bus to the same memory location repeatedly. Therefore, a memory area with the capacity of multiple frames can be used to transfer frames. When one transfer is complete, the chip simply starts over. This will alleviate the several real-time issues pertaining to this system.

However, by doing so, the software is no longer aware of the status of the transfer. If a certain are is being written by the chip while the same area is being used for transmission to the teleoperation machine, the operator will perceive a tearing effect where part of the frame is new and the rest is an older frame. In order to avoid this, markers need to be placed on each frame so that the Video4Linux driver can poll the locations corresponding to these markers and prevent reading of the memory location corresponding to frame being currently transferred over the PCI bus. Implementation of this functionality is also left as future work.

Bibliography

[1] *Information technology - coding of audio-visual objects. part 2*, ISO/IEC JTC1/SC29/WG11, Doc. N45350.

[2] A. Aydn Alatan, Yucel Yemez, Uur Gudukbay, Xenophon Zabulis, Karsten Muller, idem Erolu Erdem, Christian Weigel, and Aljoscha Smolic, *Scene representation technologies for 3DTV - a survey*, IEEE Transactions on Circuits and Systems for Video Technology **17** (2007), no. 11, 1587–1605.

[3] P. Arcara and C. Melchiorri, *Control schemes for teleoperation with time delay: A comparative study*, Robotics and Autonomous Systems **38** (2002), no. 1, 4964.

[4] Isaac Asimov, *The naked sun*, 1957.

[5] S. Battiato, M. Guarnera, G. Messina, and V. Tomaselli, *Recent patents on color demosaicing*, Recent Patents on Computer Science **1** (2008), no. 3, 194–207.

[6] A. K Bejczy, *Toward advanced teleoperation in space - NASA ATOP*, Teleoperation and Robotics in Space **161** (1994).

[7] A. K Bejczy, W. S Kim, and S. C Venema, *The phantom robot: predictive displays for teleoperation with time delay*, Proceedings of the IEEE International Conference on Robotics and Automation, vol. 1, 1990, p. 546551.

[8] P. Benzie, J. Watson, P. Surman, I. Rakkolainen, K. Hopf, H. Urey, V. Sainov, and C. von Kopylow, *A survey of 3DTV displays: techniques and technologies*, IEEE Transactions on Circuits and Systems for Video Technology **17** (2007), no. 11, 1647–1658.

[9] A. Bernardino, J. Santos-Victor, M. Ferre, and M. Sanchez-Urán, *Stereoscopic Image Visualization for Telerobotics. Experiments with Active Binocular Cameras*, Advances in Telerobotics (2007), 77–90.

[10] A. Bourge and C. Fehn, *Representation of auxiliary video and supplemental information*, ISO/IEC JTC1/SC29/WG11 FCD, 23002–3.

[11] Daniel P. Bovet and Marco Cesati, *Understanding the linux kernel, third edition*, 3 ed., O'Reilly Media, November 2005.

[12] M.Z. Brown, D. Burschka, and G.D. Hager, *Advances in computational stereo*, IEEE Transactions on Pattern Analysis and Machine Intelligence **25** (2003), no. 8, 993–1008.

[13] C. Cruz-Neira, D.J. Sandin, T.A. DeFanti, R.V. Kenyon, and J.C. Hart, *The CAVE: audio visual experience automatic virtual environment*, Communications of the ACM **35** (1992), no. 6, 64–72.

[14] GE Fanuc, *Switched ethernet latency analysis.*

[15] M. Ferre and R. Aracil, *Interfaces for telerobotics studies on stereoscopic vision*, Proceedings of the 15th IFAC Triennial World Congress, Barcelona (Spain), pp. 1315–1320.

[16] Manuel Ferre, Rafael Aracil, and Manuel Navas, *Stereoscopic video images for telerobotic applications*, Journal of Robotic Systems **22** (2005), no. 3, 131–146.

[17] S.B. Gokturk, H. Yalcin, and C. Bamji, *A Time-Of-Flight Depth Sensor-System Description, Issues and Solutions*, Proceedings of the 2004 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'04) Volume 3-Volume 03, IEEE Computer Society, 2004, p. 35.

[18] Yong Gu, Dora E Angelaki, and Gregory C DeAngelis, *Neural correlates of multisensory cue integration in macaque MSTd*, Nature Neuroscience **11** (2008), no. 10, 1201–1210.

[19] B. K. Gunturk, J. Glotzbach, Y. Altunbasak, R. W. Schafer, and R. M. Mersereau, *Demosaicking: Color filter array interpolation in single chip digital cameras*, IEEE Signal Processing Magazine **22** (2005), no. 1, 44–54.

[20] K. S Hale and K. M Stanney, *Effects of low stereo acuity on performance, presence and sickness within a virtual environment*, Applied Ergonomics **37** (2006), no. 3, 329339.

[21] B.G. Haskell, A. Puri, and A.N. Netravali, *Digital video: an introduction to MPEG-2*, Kluwer Academic Publishers, 1997.

[22] M. Hebert and E. Krotkov, *3-D measurements from imaging laser radars: how good are they?*, IEEE/RSJ International Workshop on Intelligent Robots and Systems' 91.'Intelligence for Mechanical Systems, Proceedings IROS'91, 1991, pp. 359–364.

[23] NS Holliman, *3D display systems.*, (2006).

[24] B. Holveck and H. Mathieu, *Infrastructure of the GrImage experimental platform: the video acquisition part*, Tech. report, Citeseer, 2004.

[25] D.C. Hutchison, *Introducing DLP® 3-D TV*, 2008.

[26] P. Kauff and O. Schreer, *An immersive 3D video-conferencing system using shared virtual team user environments*, Proceedings of the 4th international conference on Collaborative virtual environments, ACM, 2002, pp. 105–112.

[27] L. Kaufman, *Sight and mind: An introduction to visual perception*, Oxford University Press New York, 1974.

[28] F. L Kooi and A. Toet, *Visual comfort of binocular and 3D displays*, Displays **25** (2004), no. 2-3, 99108.

[29] Peter M. Kuhn, Georg Diebel, Stephan Herrmann, Andreas Keil, Hubert Mooshofer, Andre Kaup, Robert M. Mayer, and Walter Stechele, *Complexity and PSNR comparison of several fast motion estimation algorithms for MPEG-4*, vol. 3460, SPIE, 1998, pp. 486–499.

[30] M. S Landy, L. T Maloney, E. B Johnston, and M. Young, *Measurement and modeling of depth cue combination: in defense of weak fusion*, Vision research **35** (1995), no. 3, 389412.

[31] Nuo Li and Dora E. Angelaki, *Updating visual space during motion in depth*, Neuron **48** (2005), no. 1, 149–158.

[32] L. Lipton, *Foundations of stereoscopic cinema*, New York (1982).

[33] G. Litos, X. Zabulis, and G. Triantafyllidis, *Synchronous Image Acquisition based on Network Synchronization*, Proceedings of the 2006 Conference on Computer Vision and Pattern Recognition Workshop, IEEE Computer Society, 2006, p. 167.

[34] Yanhong Liu, S. Chakraborty, Wei Tsang Ooi, A. Gupta, and S. Mohan, *Workload characterization and cost-quality tradeoffs in mpeg-4 decoding on resource-constrained devices*, (2005), 129 – 134.

[35] S. Livatino, G. Muscato, and F. Privitera, *Stereo viewing and virtual reality technologies in mobile robot teleguide*, Robotics, IEEE Transactions on **25** (2009), no. 6, 1343 –1355.

[36] D. Marr and T. Poggio, *A computational theory of human stereo vision*, Proceedings of the Royal Society of London. Series B, Biological Sciences **204** (1979), no. 1156, 301328.

[37] D.F. McAllister, *Display Technology: Stereo & 3D Display Technologies.*

[38] P. Merkle, A. Smolic, K. Muller, and T. Wiegand, *Efficient prediction structures for multiview video coding*, IEEE Transactions on circuits and systems for video technology **17** (2007), no. 11, 14611473.

[39] M. Minsky, *Telepresence*, Omni **2** (1980), no. 9, 45–52.

[40] K. T. Mullen, *The contrast sensitivity of human colour vision to red-green and blue-yellow chromatic gratings.*, The Journal of Physiology **359** (1985), no. 1, 381.

[41] Karsten Muller, Philipp Merkle, and Thomas Wiegand, *Compressing Time-Varying visual content*, IEEE Signal Processing Magazine **24** (2007), no. 6, 58–65.

[42] J. Mulligan, X. Zabulis, N. Kelshikar, and K. Daniilidis, *Stereo-based environment scanning for immersive telepresence*, IEEE Transactions on Circuits and Systems for Video Technology **14** (2004), no. 3, 304–320.

[43] Jacob W. Nadler, Dora E. Angelaki, and Gregory C. DeAngelis, *A neural representation of depth from motion parallax in macaque visual cortex*, Nature **452** (2008), no. 7187, 642–645.

[44] Shojiro Nagata, *How to reinforce perception of depth in single two-dimensional pictures*, (1991), 527–545.

[45] Junichi Nakamura, *Image sensors and signal processing for digital still cameras*, CRC Press, 2006.

[46] H. M. Ozaktas, *Three-Dimensional television: Capture, transmission, display*, Springer Berlin Heidelberg, December 2009.

[47] Chad Fogg; Didier J. LeGall; Joan L. Mitchell; William B. Pennebaker, *MPEG video compression standard*, 1 ed., Springer, October 1996.

[48] William B. Pennebaker and Joan L. Mitchell, *JPEG still image data compression standard*, Kluwer Academic Publishers, 1992.

[49] Colin Perkins, *RTP: audio and video for the internet*, Addison-Wesley Professional, June 2003.

[50] G. F Poggio and T. Poggio, *The analysis of stereopsis*, Annual Review of Neuroscience **7** (1984), no. 1, 379412.

[51] B. Rogers and M. Graham, *Motion parallax as an independent cue for depth perception*, Perception **8** (1979), no. 2, 125–34.

[52] J.P. Rolland, L. Davis, and Y. Baillot, *A survey of tracking technology for virtual environments*, Fundamentals of wearable computers and augmented reality (2001), 67–112.

[53] K. Ruffo and P. Milgram, *Effect of stereographic + stereovideo 'tether' enhancement for a peg-in-hole task*, Systems, Man and Cybernetics, 1992., IEEE International Conference on, 18-21 1992, pp. 1425 –1430 vol.2.

[54] U. Schnars and W.P.O. J
"uptner, *Digital recording and reconstruction of holograms in hologram interferometry and shearography*, Applied optics **33** (1994), no. 20, 4373–4377.

[55] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, *RTP: A Transport Protocol for Real-Time Applications*, RFC 3550 (Standard), July 2003.

[56] S.M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski, *A comparison and evaluation of multi-view stereo reconstruction algorithms*, Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Volume 1, IEEE Computer Society, 2006, p. 528.

[57] Sarah Sharples, Sue Cobb, Amanda Moody, and John R. Wilson, *Virtual reality induced symptoms and effects (vrise): Comparison of head mounted display (hmd), desktop and projection display systems*, Displays **29** (2008), no. 2, 58 – 69, Health and Safety Aspects of Visual Displays.

[58] J. Smed, T. Kaukoranta, and H. Hakonen, *Aspects of networking in multiplayer computer games*, The Electronic Library **20** (2002), no. 2, 87–97.

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

[60] Elena Stoykova, A. Aydn Alatan, Philip Benzie, Nikos Grammalidis, Sotiris Malassiotis, Joern Ostermann, Sergej Piekh, Ventseslav Sainov, Christian Theobalt, Thangavel Thevar, and Xenophon Zabulis, *3-D Time-Varying scene capture technologies - a survey*, IEEE Transactions on Circuits and Systems for Video Technology **17** (2007), no. 11, 1568–1586.

[61] T. Svoboda, D. Martinec, and T. Pajdla, *A convenient multicamera self-calibration for virtual environments*, Presence: Teleoperators & Virtual Environments **14** (2005), no. 4, 407–422.

[62] G.K. Wallace, *The JPEG still picture compression standard*, IEEE Transactions on Consumer Electronics **38** (1992), no. 1, xviii–xxxiv.

[63] J. Wann, S. Rushton, and M. Mon-Williams, *What's wrong with your head mounted display*, CyberEdge Journal **5** (1993).

[64] Andrew J. Woods, Tom Docherty, and Rolf Koch, *Image distortions in stereoscopic video systems*, vol. 1915, SPIE, 1993, pp. 36–48.

[65] I. Yamaguchi, T. Matsumura, and J. Kato, *Phase-shifting color digital holography*, Optics Letters **27** (2002), 1108–1110.

[66] X. Zabulis, JP Barreto, N. Kelshikar, R. Molana, and K. Daniilidis, *Volumetric multi-camera scene acquisition with partially metric calibration for wide-area tele-immersion*, image **3**, no. 54, 2.

[67] Jonathan Zittrain, *The future of the internet–And how to stop it*, 1 ed., Yale University Press, April 2008.

Linux Scheduler

A

The scheduler for Linux 2.6 is much more sophisticated than earlier versions and is highly scalable (O(1)) with the number of processes. Most importantly, it differentiates between interactive, batch and real-time processes. It is a priority based preemptive scheduler. Higher priority processes can preempt lower priority processes i.e. when a higher priority process becomes runnable, it can replace the currently running process. Moreover, Linux 2.6 is a preemptive kernel which means that a process running in kernel mode (i.e. some kernel module is servicing its request), it can be preempted by another process while it is in the middle of a kernel function [11]. Priorities of conventional processes vary from 100(highest priority) to 139. Further, priority of a process affects the time quantum assigned to it:

$$\begin{aligned} Q_B &= (140 - P_S) * 20 \text{ if } P_S < 120 \\ &= (140 - P_S) * 5 \text{ if } P_S \geq 120 \end{aligned}$$

Where Q_B is the base(initial) time quantum(in ms) and P_S is static priority. Note that the kernel modules servicing a request inherit the priority of the calling process. Interactivity of processes is determined by their behavior. If a process waits for I/O from a kernel module, such as waiting for a keystroke, it is considered interactive and it is considered important to reduce the latency for such processes. While the priority of a process specified by nice or renice contributes to its initial or static priority, the scheduler calculates a dynamic priority based on the behavior of the processes. If a process sleeps a lot because of I/O calls (interactive), its dynamic priority (P_D) is raised according to:

$$P_D = max(100,\ min(P_S - Bonus + 5,\ 139))$$

Here the *Bonus* is calculated based on the time spent sleeping for I/O. Further, a process is considered 'interactive' if

$$P_D \leq \frac{3 * P_S}{4} + 28$$

Therefore, it is much easier for process with high static priority to be considered interactive. Process starvation can occur when higher priority processes lock out the lower priority processes from getting any CPU time. In order to avoid this, the scheduler keeps two lists of runnable processes: active and expired processes. Active processes are those which have not yet exhausted their time quantum. Expired processes are those which have finished their time quantum and are not run until all active processes expire. This is when the interactivity criteria comes in. Active batch processes that have finished their time quanta are expired. On the other hand, active interactive processes remain active because the scheduler refills their time quanta. Active interactive processes are

moved to the expired list only when the eldest expired process has waited for long time or if one of the expired processes has higher priority. This ensures that the processes are not starved while reducing latency for interactive processes.

Processes can be run as normal processes (discussed above), using the round-robin scheduling policy or with FIFO scheduling. Round-robin and FIFO scheduling processes have priorities ranging from 1(highest priority) to 99(lowest priority). Notice that this range is higher than normal processes (100 to 139). FIFO and round-robin processes are considered real-time processes and are always kept active. Further, a real-time is replaced with another process only when:

- It is preempted by a higher priority real-time process.
- It is put to sleep because it performed a blocking operation (called a kernel function).
- It is stopped or killed.
- It is a round-robin real-time process has used up its time quantum.
- It relinquishes the CPU on its own.

Further, the nice or setpriority system calls when used for a round-robin process priority change the length of its time quanta rather than its priority. The scheduling policy and priority(of real-time processes) can be changed with the sched_setparam and sched_setscheduler system calls or with the chrt command.