

# Tele-Immersive Display with Live-Streamed Video

TANG Wai-Kwan

A Thesis Submitted in Partial Fulfillment  
of the Requirements for the Degree of  
Master of Philosophy  
in  
Computer Science and Engineering

©The Chinese University of Hong Kong  
July, 2001

The Chinese University of Hong Kong holds the copyright of this thesis. Any person(s) intending to use a part or the whole of the materials in this thesis in a proposed publication must seek copyright release from the Dean of the Graduate School.



# Tele-Immersive Display with Live-Streamed Video

submitted by

**TANG Wai-Kwan**

for the degree of Master of Philosophy  
at The Chinese University of Hong Kong

## Abstract

Tele-immersion is a rapidly growing technology which allows geographically separated users to collaborate in a shared virtual environment. Video systems in those tele-immersion applications, however, usually suffer from a limited field-of-view. To capture wide field-of-view video, specially designed video systems are required.

In this thesis, we present the techniques for the real-time construction of live panoramic video from multiple live video streams obtained by normal CCD video cameras. A system framework for tele-immersive applications is designed. In our approach, we first obtain sample video frames from the video cameras. Using techniques of generating image mosaic, the video frames are combined together to form a large field of view image. We obtain a list of triangles with texture coordinates. In the running time, the live video streams are used as texture map which are rendered into a live panoramic video stream. The generated video is then projected onto an immersive display.

Our system framework can allow many potential tele-immersive applications. For example, almost all current video conferencing applications are

limited to a small field of view video. Using our system framework, instead of seeing only a talking head, the users can get the feeling of immersing into a virtual conference room. Virtual control room, immersive remote navigation, virtual laboratory, tele-medicine are some possible directions which can gain benefit from our system framework.

## 內容摘要

遠距離沉浸 (Tele-immersion) 是一門快速發展中的新技術，它使分佈於世界各地的使用者能夠在一個共享的虛擬環境中互相合作，但這些遠距離沉浸應用中的視像系統卻有很大的視野限制，唯有用上特別設計的視像系統，才可以擷取擴闊視野的視像。

本論文中，我們提出一種即時全境視像的實時構造技術，而那些即時視像訊號是由普通的電荷耦合器 (CCD) 式攝取機中擷取的。我們設計了一個遠距離沉浸的系統架構。我們首先由攝像機擷取單格視像，再用上圖像接合技術，以製成一幅擴闊視野的圖像。由此，我們可以得到一串有材質貼圖紋理座標的三角形。在執行時，即時的全景視像便可由那些即時視像訊號的材質貼圖 (Texture Map) 繪製出來，再投射上一個沉浸的顯示器 (Immersive Display)，用者便有置身虛擬的環境當中的感覺。

多種遠距離沉浸應用可以建構在我們的系統架構，舉例來說，現有的視像會議應用多是限制於狹窄視野的視像。若是用上我們的系統架構，視像會議將變成虛會議室。除此之外，一些實際上的應用，包括虛擬控中心、搖距漫遊、虛擬實驗室、遙距醫療等也可以由我們的研究成果上有所得益。

# Acknowledgment

Lots of people have contributed and supported this work. First of all, I would like to express my heartfelt gratitude to my supervisors, Dr. Pheng-Ann Heng and Dr. Tien-Tsin Wong, for their insights and great ideas about this work. This work would not be completed without their guidance and support.

My thesis readers, Dr. Han-Qiu Sun and Dr. Kin-Hong Wong, have spent time and effort on reading the work. They provided me lots of valuable and constructive comments. I want to give my thanks to them.

I would also like to thank my colleagues: Mr. Michael Fung, Mr. Napoleon Lee, Mr. Raymond Chung, Mr. Yim-Pan Chui, Mr. Philip Fu, Mr. Keung-Tat Lee and other members in the graphics group. Although we are working on different research topics, they gave me valuable suggestions and support. Also, many thanks to the helpful system administrators in the Department of Computer science and Engineering.

Most importantly, I would like to give my sincere thanks to my family for their love, care and encouragement which give me the incentive to finish this work.

# Contents

<b>Abstract</b>	<b>i</b>
<b>Acknowledgement</b>	<b>iii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Applications . . . . .	3
1.2 Motivation and Goal . . . . .	6
1.3 Thesis Outline . . . . .	7
<b>2 Background and Related Work</b>	<b>8</b>
2.1 Panoramic Image Navigation . . . . .	8
2.2 Image Mosaicing . . . . .	9
2.2.1 Image Registration . . . . .	10
2.2.2 Image Composition . . . . .	12
2.3 Immersive Display . . . . .	13
2.4 Video Streaming . . . . .	14
2.4.1 Video Coding . . . . .	15
2.4.2 Transport Protocol . . . . .	18
<b>3 System Design</b>	<b>19</b>
3.1 System Architecture . . . . .	19
3.1.1 Video Capture Module . . . . .	19

3.1.2	Video Streaming Module . . . . .	23
3.1.3	Stitching and Rendering Module . . . . .	24
3.1.4	Display Module . . . . .	24
3.2	Design Issues . . . . .	25
3.2.1	Modular Design . . . . .	25
3.2.2	Scalability . . . . .	26
3.2.3	Workload distribution . . . . .	26
<b>4</b>	<b>Panoramic Video Mosaic</b>	<b>28</b>
4.1	Video Mosaic to Image Mosaic . . . . .	28
4.1.1	Assumptions . . . . .	29
4.1.2	Processing Pipeline . . . . .	30
4.2	Camera Calibration . . . . .	33
4.2.1	Perspective Projection . . . . .	33
4.2.2	Distortion . . . . .	36
4.2.3	Calibration Procedure . . . . .	37
4.3	Panorama Generation . . . . .	39
4.3.1	Cylindrical and Spherical Panoramas . . . . .	39
4.3.2	Homography . . . . .	41
4.3.3	Homography Computation . . . . .	42
4.3.4	Error Minimization . . . . .	44
4.3.5	Stitching Multiple Images . . . . .	46
4.3.6	Seamless Composition . . . . .	47
4.4	Image Mosaic to Video Mosaic . . . . .	49
4.4.1	Varying Intensity . . . . .	49
4.4.2	Video Frame Management . . . . .	50
<b>5</b>	<b>Immersive Display</b>	<b>52</b>

5.1	Human Perception System . . . . .	52
5.2	Creating Virtual Scene . . . . .	53
5.3	VisionStation . . . . .	54
5.3.1	F-Theta Lens . . . . .	55
5.3.2	VisionStation Geometry . . . . .	56
5.3.3	Sweet Spot Relocation and Projection . . . . .	57
5.3.4	Sweet Spot Relocation in Vector Representation . . . . .	61
<b>6</b>	<b>Video Streaming</b>	<b>65</b>
6.1	Video Compression . . . . .	66
6.2	Transport Protocol . . . . .	66
6.3	Latency and Jitter Control . . . . .	67
6.4	Synchronization . . . . .	70
<b>7</b>	<b>Implementation and Results</b>	<b>71</b>
7.1	Video Capture . . . . .	71
7.2	Video Streaming . . . . .	73
7.2.1	Video Encoding . . . . .	73
7.2.2	Streaming Protocol . . . . .	75
7.3	Implementation Results . . . . .	76
7.3.1	Indoor Scene . . . . .	76
7.3.2	Outdoor Scene . . . . .	78
7.4	Evaluation . . . . .	78
<b>8</b>	<b>Conclusion</b>	<b>83</b>
8.1	Summary . . . . .	83
8.2	Future Directions . . . . .	84
<b>A</b>	<b>Parallax</b>	<b>86</b>

# List of Tables

5.1	Contribution of Human Senses . . . . .	53
7.1	Comparison of video codec . . . . .	74
7.2	Network properties . . . . .	75

# List of Figures

1.1	Tele-Immersive Applications . . . . .	2
1.2	Lifter Crane . . . . .	4
1.3	Virtual lifter crane . . . . .	5
1.4	Robots . . . . .	5
1.5	Modified Sojourner . . . . .	6
2.1	Various immersive displays . . . . .	13
3.1	System Architecture . . . . .	20
3.2	Fisheye lens and its image . . . . .	20
3.3	Camera Group . . . . .	21
3.4	Camera Arrangement . . . . .	22
3.5	VisionStation . . . . .	25
4.1	Specially designed camera cluster . . . . .	29
4.2	Processing Pipeline . . . . .	32
4.3	Pinhole Camera Model . . . . .	34
4.4	Illustration of Perspective Projection . . . . .	34
4.5	Different types of lens distortion . . . . .	36
4.6	Calibration Board . . . . .	38
4.7	Common reference coordinate system . . . . .	40
4.8	Stitching Hierarchy . . . . .	47

4.9	A mosaic formed by seamlessly combining 7 images . . . . .	48
4.10	Video capture without synchronization . . . . .	50
5.1	Relocation of Sweet Spot . . . . .	54
5.2	F-Theta lens . . . . .	56
5.3	VisionStation . . . . .	57
5.4	Projection on VisionStation . . . . .	58
5.5	Solutions of $z_s$ . . . . .	59
5.6	Coordinates System of Projector . . . . .	60
5.7	Projection for VisionStation . . . . .	62
6.1	Jittering . . . . .	68
7.1	Video capture module . . . . .	72
7.2	User interface of the video capture program . . . . .	73
7.3	camera arrangement for indoor scene . . . . .	76
7.4	Correspondence is identified . . . . .	77
7.5	Input Images are stitched together . . . . .	77
7.6	Stitched image is warped . . . . .	78
7.7	Warped image is projected onto VisionStation . . . . .	79
7.8	Outdoor image sources . . . . .	79
7.9	Outdoor image sources . . . . .	80
7.10	Outdoor image sources . . . . .	80
7.11	Capturing at the dawn . . . . .	81
A.1	Parallax of two cameras . . . . .	87

# Chapter 1

## Introduction

Advance in today's computer technologies open up new challenging direction in many areas such as computer graphics and virtual reality. In particular, tele-immersion is one of the emergent areas where extensive research is now working on. Tele-immersion enables users at geographically distributed sites to collaborate in a simulated, shared virtual environment in a real-time fashion. And the ultimate goal of tele-immersion would be the generation of a virtual environment which provides users the full sensory perception as actually present in the real world and allows users to interact with either the virtual environment and/or the other users. Under this concept, its great potentials are still not yet fully released. In fact, tele-immersion is still in its infancy and that ultimate goal may not be possible in near future. It does, however, allow large room to explore.

Currently, the major areas of tele-immersion applications are:

- *Collaborative Working*

Tele-conferencing allows the carrying of meetings around the world. Also, multiple participants can work in a shared virtual environment to do CAD design or prototyping.

- *Simulation and Training*

Pilot trainee can use flight simulators to enhance their learning. This can reduce the danger of premature flight training and can save money. Military school can train soldiers inside a virtual battle field.

- *Entertainment*

People can play video games together, interacting with each other in a common virtual world.

- *Tele-education and Tele-medicine*

Students may remotely participate in virtual classrooms and patients can receive medical diagnosis from doctors.

- *Remote Controlling*

Space exploration, manufacturing, surveillance, etc. can use tele-immersion system to control remotely the machines as actually present at that location.



(a) Flight Simulator



(b) Tele-Medicine

Figure 1.1: Tele-Immersive Applications

Tele-immersion technologies do not limit in the above applications and it allows more varieties of applications. It is a cross disciplinary research field that incorporates many of the techniques and algorithms from areas such as computer graphics, vision, user interface, video coding, and networking. There are significant research and practical challenges of building such tele-immersive application.

In this research, we develop a framework for live video based tele-immersion applications. It uses an integrated approach coming from three main fields. In the context of computer vision, it is about the data acquisition and reconstruction. In the context of computer graphics, it is about the realistic and interactive display of data. And in the context of networking, it is about the real-time, low latency, low cost and high reliability of data transmission across the network.

The highlights of the system framework are:

1. Mosaicing live-video stream.
2. Real-time video streaming using off-the-shelf components.
3. Displaying of panoramic video using immersive technology.

## 1.1 Applications

Under our proposed framework, a variety of tele-immersion applications can be developed. The most straightforward application is extending the current video-conferencing model. Most video-conferencing applications in the market allow only a face-to-face video conferencing through the flat display panel. But using our framework, a person can "immerse" into the conference room distant away. Other examples might be to participate or consult in a surgery from a remote location (tele-medicine), or to remotely participate in a virtual classroom.

Another group of applications is virtual control room. In many situations, such as controlling industrial machines, cruisers, tanks, submarines or airplanes, it is necessary to have a complete understanding of the surroundings. However, the observable field of view from inside the control chamber is

often limited due to the fact that structures often blocking part of the view. Using other technologies like radar or infra-red detection, more information of the surroundings can be obtained. In many cases, a complete view of the surroundings is still the most important source of information and cannot be substituted.

As an example application, construction machines like lifter crane can employ our system. Inside a crane operator cabin, the visibility of looking out is very limited (Figure 1.2b).

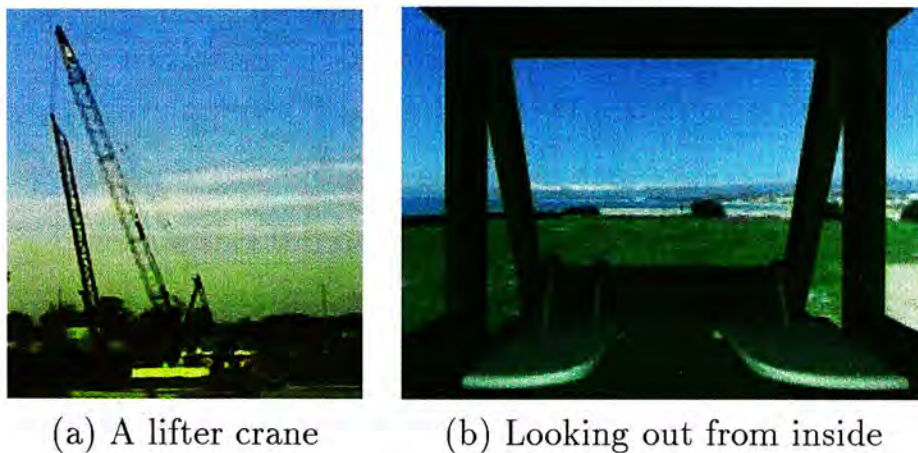


Figure 1.2: Lifter Crane

The operator needs to have a full understanding of the environment when controlling the movement of the crane. If some part of the environment is invisible, it is dangerous as the crane may hit any buildings and machines nearby. To reduce the danger, providing the operator with enough environment information is useful. By mounting cameras outside the operator cabin, full information of the surroundings can be provided to the operator. And using our system, which can surely help the operators to understand the information. In the future, no operator might be required to work on the crane. The operator might be situated in a virtual environment as of the crane and could control the crane remotely.

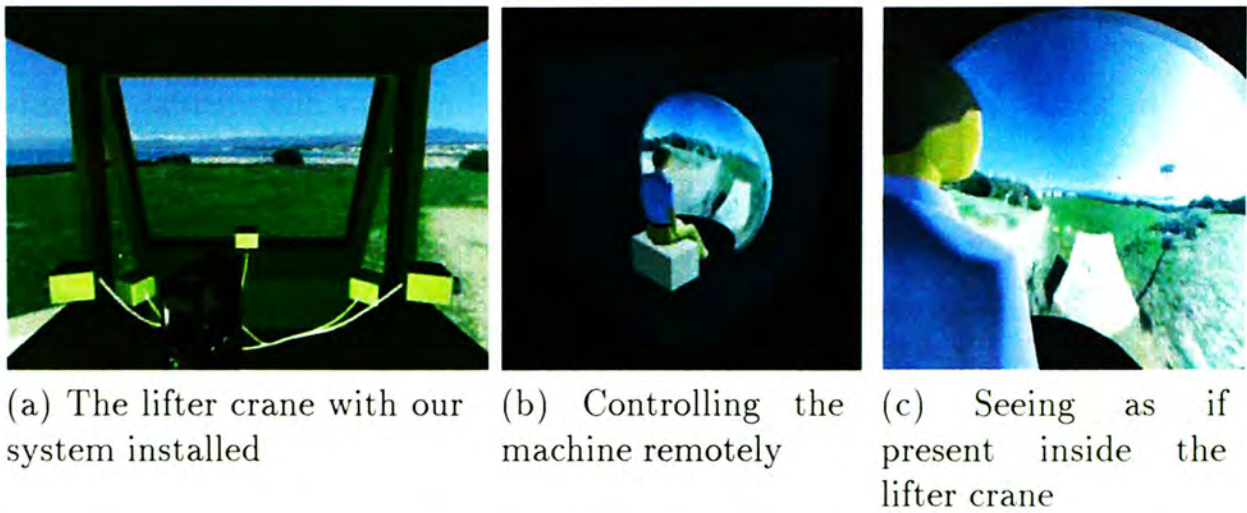


Figure 1.3: Virtual lifter crane

Besides, our framework can be used in real-time remote navigation system. For example, by mounting a cluster of cameras onto a robot, it allows any remote users viewing the surroundings as seen from the view point of the robot. This kind of system is especially suitable in situation where a site is unsafe for human to enter, such as the area with fire/nuclear disaster (Figure 1.4), deep in ocean or out in the space, etc.

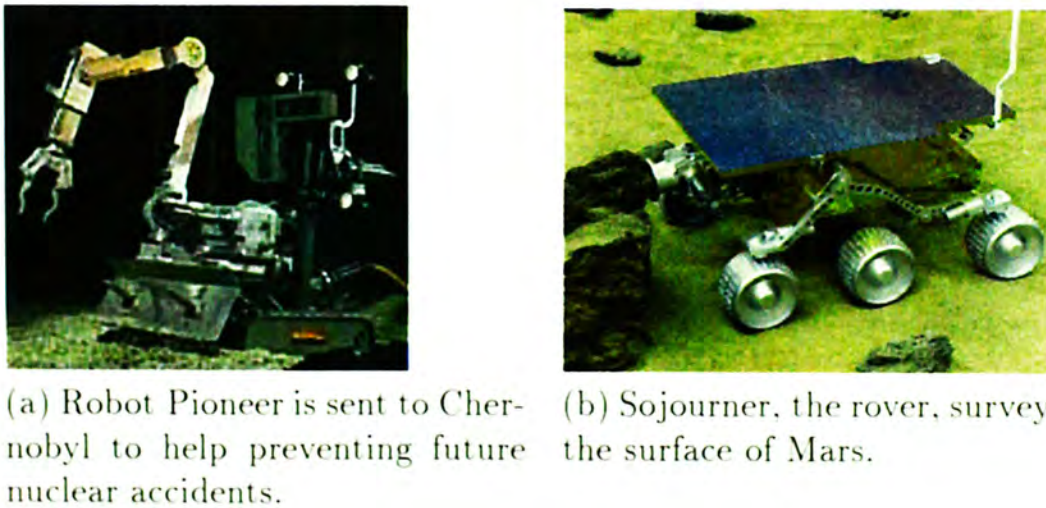


Figure 1.4: Robots

As an imaginary example, consider the Mars exploration occurred in July 1997. Suppose Sojourner, the rover, was mounted with the video capture

devices of our tele-immersion system as show in Figure 1.5a. The capture devices capture the surroundings and then the signals are sent back to the Earth. People sitting remotely in front of a immersive display on the Earth can navigate on the red planet as really there. In future Mars exploration, our system can be employed which can not only provide an interesting navigation experience but also help understanding the surroundings.



(a) Imaginary Sojourner with our system installed



(b) Navigating Mars on Vision-Station

Figure 1.5: Modified Sojourner

## 1.2 Motivation and Goal

The demand for tele-immersive applications is increasing. However, current tele-immersive technologies are still not mature enough to serve the increasing need. This motivates us to develop a system framework for various classes of tele-immersive applications. Thus, the goal of this project is to design and develop a framework for tele-immersive applications. As stated before, the framework highlights on the following aspects:

1. Mosaicing live-video streams.
2. Real-time video streaming using off-the-shelf components.

3. Displaying of panoramic video using immersive technology.

## 1.3 Thesis Outline

In Chapter 2, a survey on current development and technologies of the related areas will be given. This includes image mosaicing, video streaming, immersive display and etc.

In Chapter 3, we introduce the framework of our tele-immersive system. It provides an overview of how and why it is suitable for a tele-immersive system.

Our system involves a number of aspects which are described in detail in Chapter 4 to 6. Chapter 4 describes the mosaic techniques in combining the multiple video streams into a single panoramic video. Chapter 5 explicates how immersive display is used in our system to enhance the user perception in utilizing the system. And in Chapter 6, the considerations in achieve video streaming across the network will be given.

After discussing the detail of our framework and each of the components, experimental results and comparisons are given in Chapter 7.

Finally, Chapter 8 summarizes this project and discuss the improvements and possible future works.

## Chapter 2

# Background and Related Work

Our research involves a number of different technologies, ranging from video capturing, networking, to immersive display. We will first give a glance of related systems. And then in each of the subsequent sections, a specific related area is reviewed.

### 2.1 Panoramic Image Navigation

Panoramic image navigation systems, such as QuickTime VR[1] and IPIX[2], allow users to navigate around a virtual visual environment. Unlike ordinary photographs, they allow interaction like panning, zooming and tilting around a node. And unlike virtual 3D world, they usually use images as input and so provide higher realism. To achieve realistic virtual environment walk-through, one can first capture panoramic images at different nodes. The nodes are then linked together. For each node, the user can click their mouse/cursor on a doorway, for example, and be instantly transported to the next node, which could then to the next node, and so on. These systems, however, have a major limitation - the virtual environment is static and has to be prepared offline.

Some other systems allow the playback of dynamic contents. These include the Omnimax and the IMAX dome theater, which can provide audience with panoramic video. It is closer to tele-immersion usages, but they suffer from the problem that the content has to be prepared offline and using sophisticated video capturing devices. Very few systems can capture video panoramic live video and allow the playback in real-time.

All the systems mentioned require the input of panoramic image or panoramic video stream. To capture live panoramic video streams, most of them require the use of specially designed camera or camera cluster. Using fish-eye lenses, it is possible to capture video as wide as  $180^\circ$  field of view. It is expensive and also the captured images are highly distorted. Nayar[3, 4] has developed a polycamera, which is built from a cluster of cameras, which combines the multiple video streams from the cameras to form a  $180^\circ$  panoramic video in real-time. Joshua[5] has developed a compact panoramic stereo camera which uses parabolic mirrors and it is capable to produce  $360^\circ$  panoramic depth maps at interactive frame rate. The video panoramas project[6] in Stanford University can create real-time  $360^\circ$  video from a ring of carefully aligned cameras. As the configuration of the camera clusters is known in advance, it is relatively easier to combine different cameras view together. For randomly placed cameras, there is still no solution for doing mosaicing on the video streams in real-time.

## 2.2 Image Mosaicing

Instead of taking video streams as input, there are quite a number of works that compose images to form panoramic mosaic. The construction of mosaic

images is an active area of research in recent years. This image-based rendering problem combines two complementary fields: computer vision and computer graphics [7]. In computer graphics world, mosaics are usually used as environment maps which are static background of synthetic scenes. Mounting such environment maps on a cylindrical or spherical surface [1, 8, 9] and allowing the users to pan, rotate or zoom on the surface, users can feel the sense of immersion into the virtual scene. Image mosaicing can be considered as two subproblems. Image registration is to find how the images are aligned and image composing is to combine seamlessly based on the alignment.

### 2.2.1 Image Registration

Image registration is the core of image mosaicing which it match two or more images. It is a central issue for a variety of problem in image processing[10] such as object recognition, motion tracking, matching stereo images for depth reconstruction, etc. There are many solutions or at least partial solutions to the general image mosaicing problem. Most of them can be roughly classified as one of the three approaches: phase correlation, transformation recovery and video sweeping.

- *Phase Correlation*

Phase correlation is a straightforward approach to image mosaicing based on the image properties in frequency domain. First proposed by Kuglin and Hines[11] in 1975, the algorithm uses 2D Fourier transform and computes the displacement between two images from the phase of their cross power spectrum. This algorithm is scene-independent and accurate to within one pixel for image differing by a pure 2 dimensional translation. Instead of finding the displacement, other methods[12, 13] find the rotation and scale for image registration. These algorithms,

however, are quite sensitive to noise and also require the overlap extent to occupy a significant portion of the images (e.g. at least 50%).

- *Transformation Recovery*

Transformation recovery is a more intuitive approach to image mosaicing. This approach makes use of the fact that two spatially neighboring images are related to each other by a homography transformation. Having recovered the homography transformation, the images can be aligned together.

To recover the homography transformation, feature based [14, 15] methods rely on the detection of image features, with which the camera motion can be computed. In the absence of distinctive features, automatic feature extraction methods usually failed. The other way to recover the homography transformation is to iteratively adjust the camera motion parameters. Szeliski [16, 17] proposes the use of Levenberg-Marquardt minimization to find the transformation matrix. Together with Shum[18, 9], they parameterize the 2D mosaicing problem by using rotations and focal lengths of the camera that is moving around a fixed point. Similar work is done by Sawhney and Kumar[19].

- *Video Sweeping*

Video sweeping approach[20, 17, 21] creates panoramic mosaics by combining the scene information from the frames in a video sequence. The motion of the camera is recovered and thus the information of the video frames can be properly combined together. Peleg and Herman[20] used a manifold projection which simulates the sweeping of the scene with a plane using an one dimensional sensor array. Their method works fine for a single video strip only but usually fail for the case of multiple video strips. Sawhney, Hsu and Kumar[21] improved that by using

topology inference and local-to-global alignment.

### 2.2.2 Image Composition

After geometric corrections, the images are aligned in the registration process. These aligned images require further processing to eliminate remaining artifact. Alignment of images may be imperfect due to registration errors, incompatible model assumptions, dynamic scenes, etc. Moreover, images are usually exposed unevenly due to the changing of lighting conditions and automatic gain controls of cameras. These undesirable effects can be alleviated by the composition process.

The main problem in image composition is to present the pixels in overlapping area between two images. Finding the best border[15, 22] in the overlapping area to separate the images can eliminate remaining artifacts and help to avoid double exposure. Another way to seamless composition is to combine the overlapping area using a smooth blending function called feathering. Burt[23] proposed a multi-resolution approach for merging images. Each source image is decomposed into a set of band-pass filtered images. For each band, the component images are merged separately together using a specific weighted average function within a transition zone. This method can produce the composition of images. However, it is computationally intensive which limits it from using in real-time applications. The uneven exposure problem can be reduced by using histogram equalization[22].

## 2.3 Immersive Display

Immersive displays come in two forms. The first is the head mount display (HMD) (Figure 2.1a). The second is a very large screen, typically projected onto a wall and referred to as a spatially immersive display (Figure 2.1b,c,d). Before 1992, head mount display (HMD) is the only immersive display technology available. However, HMDs have a number of deficiencies - they separate the users from their familiar environment, rather low resolution, cumbersome, heavy and uncomfortable to wear. So in 1991, DeFanti and Sandin starts researching on spatially immersive display and they developed the CAVE Automatic Virtual Environment (CAVE)[24, 25] in 1992. CAVE (Figure 2.1b) is a projection-based virtual reality environment which uses 3D computer graphics and position tracking to immerse users inside a 3D space.

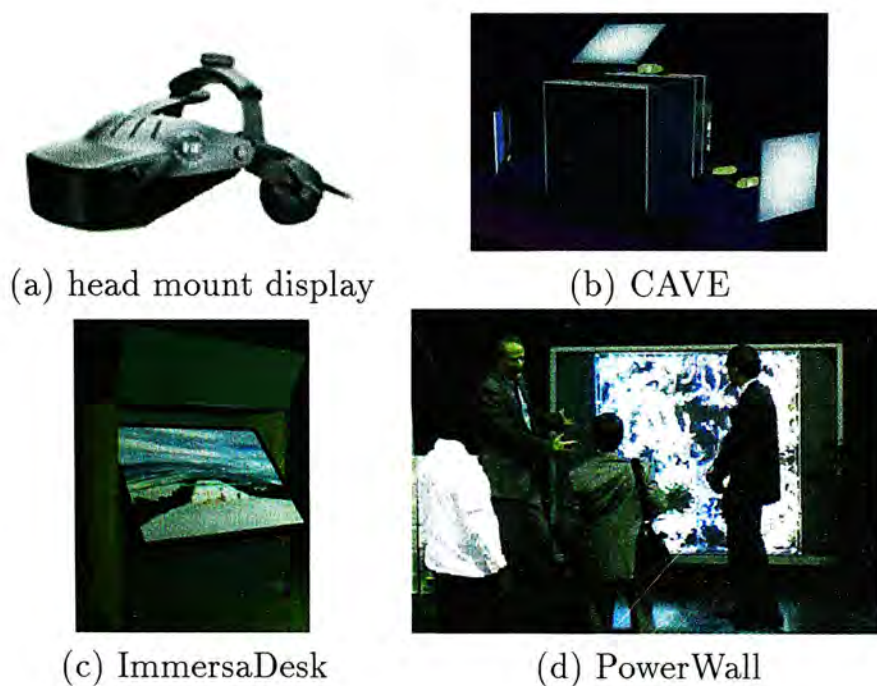


Figure 2.1: Various immersive displays

Since then, the focus becomes the building of display system from multiple screens and projectors. The problem becomes finding the relationships

between the screens and projectors so that the system can be used as if it is a simple display system. It promises very high-resolution displays with large areas of coverage using low-cost components. Such large field of view display is usually configured accurately so that the projectors are physically aligned to match nearby projectors[24, 25, 26, 27, 28, 29]. With randomly placed multiple projectors, Raskar[30] solves the problem of aligning the projected images so that they appear as a seamless one.

## 2.4 Video Streaming

Conventional playback of media files across the network requires that the entire media clip being downloaded in prior. This can take minutes or even hours of time for long video clips. For live video captures, however, it is impossible to record to a video clip first and then deliver the whole clip later. Streaming enables content to be delivered on demand to the client as a continuous flow of data with little waiting time before playback begins. The length of the video clip can be unlimited. This is the case of our live video streams. The idea of streaming video over the network has received a lot of attention recently. Products such as RealNetworks' streaming system<sup>1</sup>, Microsoft's Windows Media Technologies<sup>2</sup> and Apple's QuickTime,<sup>3</sup> are widely in use. In a heterogeneous network, bandwidth, latency and other network parameters varies. It is necessary to control the Quality of Services (QoS), which enables smooth playback of video data across the network. When designing video streaming applications[31, 32, 33], two major issues are involved: video compression techniques and transport protocols.

---

<sup>1</sup><http://www.realn networks.com/>

<sup>2</sup><http://www.microsoft.com/windows/windowsmedia/>

<sup>3</sup><http://www.apple.com/quicktime/>

### 2.4.1 Video Coding

Networks nowadays are usually bandwidth-constrained, video data must be compressed before transfer to reduce their data rates. Consider an example, suppose the pictures in a sequence are digitized as discrete grids or array with 320 pixels per raster line and 240 lines per picture, 24 bits per pixel. If the moving pictures are sent uncompressed at 24 pictures per second, the raw data rate for the sequence is about 44 Mbit/s. Transmitting 4 such video streams will result in 176 Mbit/s which will overwhelm even the very fast 155 Mbit/s ATM network. We need to compress the video stream in a way that multiple video source can be transmitted across the network at the same time.

The compression algorithm, together with its corresponding decompression algorithm, are called *codec*. To do the compression, the compression algorithm is applied that analyze the data and decide which bits of data can be removed or merged in order to minimize visible artifact. This, however, inevitably results in some loss of quality. The level of quality and fidelity after compression and decompression depends primarily on how much bandwidth is available. And in a given bandwidth, the quality of the content is largely determined by how good the codec is in dealing with a particular video dataset. In nowadays technology, the most widely used codec standard for video streaming are H.261, H.263, Motion JPEG (MJPEG), MPEG-1, MPEG-2 and MPEG-4. QuickTime, RealMedia and ASF are also widely used, but they are using proprietary codec which are not considered here. They have their own strength and weakness that suit some particular situations.

- *H.261/H.263*

H.261 is also known as  $P \times 64$  where  $P$  is an integer number meant to represent multiples of 64 Kbit/s. H.261 was targeted at teleconferencing applications and is intended for carrying video over ISDN - in particular for face-to-face video-phone applications and for video conferencing. The actual encoding algorithm is similar to that of MPEG. H.261 needs substantially less CPU power for real-time encoding than MPEG. The algorithm includes a mechanism which optimizes bandwidth usage by trading picture quality against motion, so that a quickly changing picture will have a lower quality than a relatively static picture. H.261 is thus not a constant quality codec, but it is a constant bit rate codec with variable video quality.

H.263 is a draft ITU-T standard designed for low bit rate communication. The encoding algorithm of H.263 is similar to that used by H.261 with some improvements and changes to improve performance and error recovery.

- *MPEG*

MPEG are developing standards for the bandwidth efficient transmission of video and audio. The MPEG-1 codec targets a bandwidth of 1-1.5Mbit/s offering VHS quality video at CIF ( $352 \times 288$ ) resolution and 30 frames per second. MPEG-1 requires expensive hardware for real-time encoding. While decoding can be done in software, most implementations consume a large fraction of a high-end processor. MPEG-1 does not offer resolution scalability and the video quality is highly susceptible to packet losses, due to the dependencies present in the P (predicted) and B (bi-directional predicted) frames. The B-frames also introduce latency in the encode process, since encoding the  $N$ -th frame needs access to  $(N+k)$ th frame, making it less suitable for

video conferencing.

MPEG-2 extends MPEG-1 by including support for higher resolution video and increased audio capabilities. The targeted bit rate for MPEG-2 is 4-15 Mbit/s, providing broadcast quality full screen video. It provides three types of scalability: signal-to-noise ratio (SNR), spatial and temporal, and data partitioning. Compared with MPEG-1, it requires even more expensive hardware to encode and decode. It is also susceptible to the presence of losses as for MPEG-1.

The intention of MPEG-4 is to provide a compression scheme suitable for video conferencing at a data rate less than 64 Kbit/s. The MPEG-4 framework currently being developed focuses on a language called MSDDL (MPEG-4 Syntactic Description Language). MSDDL allows applications to construct new codecs by composing more primitive components over the Internet.

- *MJPEG*

There is really no standard as "Motion JPEG" or "MJPEG" for video. Various developers have applied JPEG to individual frames of a video sequence and have called the result "MJPEG". JPEG is designed for compressing either full color or gray scale image for natural, real-world scenes. It works well on photographs, and natural scenes; not so well on documents, cartoons or line drawings. It is based on a lossy compression algorithm which use DCT-based encoding. Since it does not exploit the intra-frame coherence, less computation is required and there is little latency as compared to MPEG or H.261/H.263.

## 2.4.2 Transport Protocol

The task of transport protocol is to provide reliable, cost-effective data transport from the source machine to the destination machine, independent of the physical network in use. The Internet has two main protocols, a connection-oriented transmission control protocol(TCP) and a connectionless user datagram protocol (UDP). Other protocols base on TCP and UDP are developed for special purposes.

- *Transmission Control Protocol (TCP)* - TCP is a connection-oriented protocol, which means a connection must establish with each other before exchanging data. It provides a reliable end-to-end byte stream service.
- *User Datagram protocol (UDP)* - UDP is a simple, datagram-oriented protocol. Unlike TCP, there is no guarantee that the datagrams can reliably reach the destination.
- *Real-Time Transport Protocol (RTP)* - RTP[34] is a protocol for the transport of real-time data, including audio and video. It can be used for media-on-demand as well as interactive services such as Internet telephony.
- *Real-Time Streaming Protocol (RTSP)* - RTSP[35] is a communications protocol for control and delivery of real-time media. It defines the connection between streaming media client and server software, and provides a standard way for clients and servers from multiple vendors to stream multimedia content.

## Chapter 3

# System Design

In this chapter, we will first discuss the system architecture and the functions of each modules. The last section will describe about the system design issues.

### 3.1 System Architecture

Our system consists of four main modules as depicted in Figure 3.1. The video capture module captures the real world scenery and convert into digital video signals. The video streaming module allows the captured video to transmit across the network. The stitching module combines the different video frames of an instance to form a single panoramic frame. And finally, the display module presents the panoramic video to the viewers. The following sections illustrate each of the four modules.

#### 3.1.1 Video Capture Module

The task of the video capture module is to obtain live video streams. In order to obtain live video streams, capturing devices, such as digital camcorders or charge coupled devices (CCD), should be used. Most of these capturing devices have a very limited field of view, typically around 60 degree diagonally. To capture a wide field, there are two main solutions. The first one is

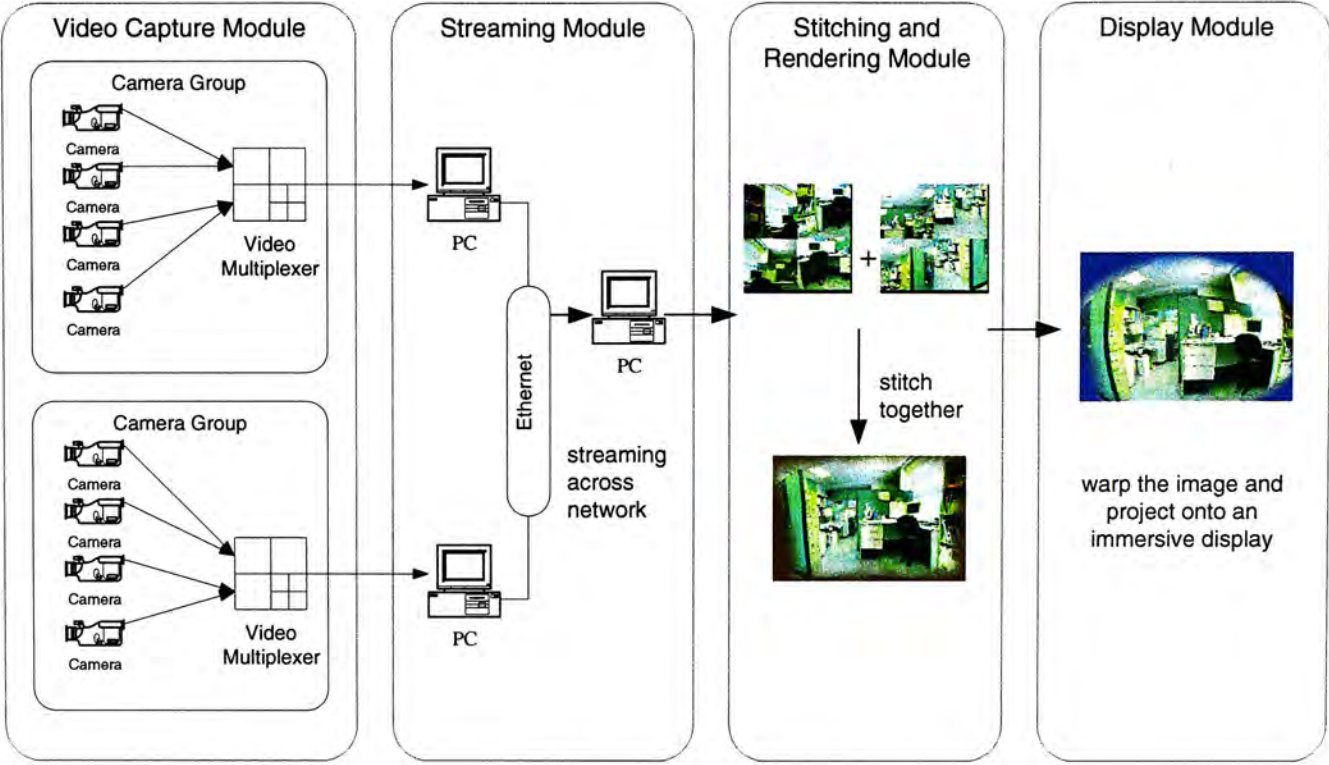


Figure 3.1: System Architecture

to use capturing devices with wide field of view. These devices usually employ lenses with very short focal length. Not counting for the more expensive price, those lenses inevitably introduce great distortion to the captured image (Figure 3.2). The other solution is to use multiple ordinary cameras, with each of them captures different part of the surroundings. The multiple video streams captured are then combined together to form a single panoramic video.



Figure 3.2: Fisheye lens and its image

In our current system, the multiple camera approach is adopted. Using

eight CCD cameras, eight PAL video signals which span roughly a hemisphere of the surrounding are captured. Two quad video mergers are used to combine the analog signals from the eight cameras into two video signals, each divided into a quadrants showing one of the cameras (Figure 3.3) These two analog video signals are digitized using video capture boards on two independent computers. The digitized video streams are then passed to the streaming module running on both computer.

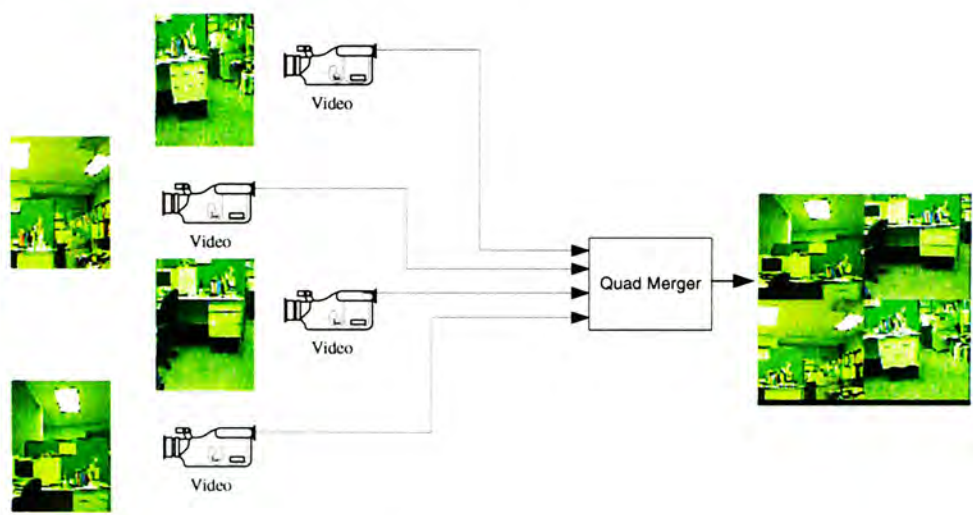


Figure 3.3: Camera Group

Combining the four analog video signals into one by the quad video mergers will unavoidably downsamples the quality of the captured signals. However, there are two main reasons for using them:

1. Each analog video stream requires one video capture board to digitize the video. Unless with very sophisticated capture board design, a single computer can have handle one video stream at a time. So for the case of 8 input video streams, eight capture boards on eight separate computers are required. It is impractical and infeasible to fulfill this hardware requirement.

2. Even the hardware requirement is not a concern, managing eight computers is difficult. The eight computers should be connected together which complicates the network configuration. And it introduces more complex problems like synchronizations.

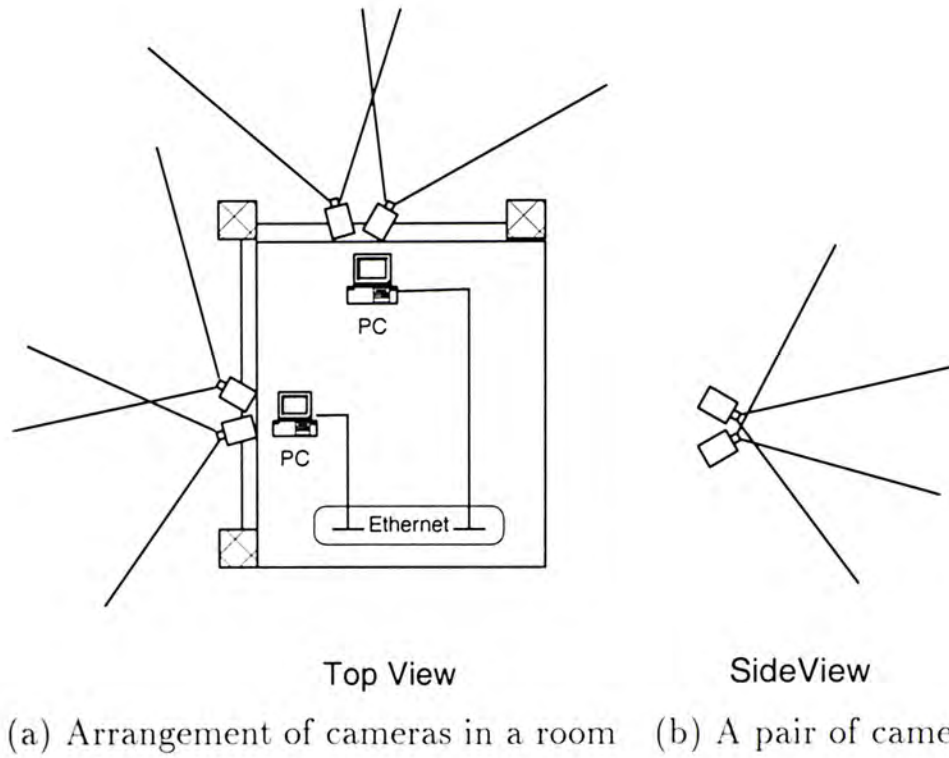


Figure 3.4: Camera Arrangement

The arrangement of the cameras is important. They are positioned in such a way that they are all pointing to different directions with part of the capturing area overlapped. These overlapping areas are important for later processing. Figure 3.4a shows the current camera arrangement of our systems. The capturing module are divided into two capturing groups. Each group has four cameras, a quad video merger and a computer. Every two cameras are fixed together as should in the Figure 3.4b. This arrangement can help minimizing the distance between the optical centers of the cameras. The two groups are located in different location of a room looking outside.

With this arrangement, it is possible to cover approximately 160 degree horizontally and 120 degree vertically. To enlarge the coverage, more capturing groups can be added.

### **3.1.2 Video Streaming Module**

The task of the video streaming module is to transmit the captured video data from the capturing groups to the stitching module. The capturing groups are located in the remote site where we want to observe. The stitching module, on the other hand, is located at the place where the viewer is. It can be very close to the remote site, or it can be located hundreds miles away. They are connected through a network. This requires the real-time network streaming.

The delivery model for our streaming module follows the simple peer-to-peer design. This consists of the following steps:

1. When the streaming starts, messages are sent to the capturing groups to start the capturing sessions.
2. Video frames captured are encoded by a video compressor.
3. The encoded data is sent across the network to the receiver.
4. The receiver gets the encoded data and decompresses it using the corresponding video decompressor.
5. The recovered video frames are passed to the stitching and rendering module.

When dealing with the streaming across the network, three main issues arise. They are the choice of video compression and decompression engine

(codec), how the data is transmitted through the network and how to manage the network loss, latency and jitter. Our system is designed in a scalable fashion that the codec and the transmission protocol can be changed and extended easily to satisfy different situations and requirements.

### **3.1.3 Stitching and Rendering Module**

Stitching Module, being the core of the whole system, combines the captured video streams together to form a panoramic video. As video is basically a sequence of images, stitching video streams can be reduced to the problem of stitching video frames of same instance together to form a panoramic video frame. This problem is similar to the image mosaics which we repeat the mosaicing process for every video instance. For the video frame sequence, the difference between consecutive frames will not be very much. And if the cameras are fixed in their relative positions, then the image mosaic process of an instance will be the same as that in the next instance. So, the image mosaicing parameters of one instance can be applied on the other instance. In our framework, the stitching process is carried out once. Using the stitching result, rendering of the panoramic image is done on every video instance.

### **3.1.4 Display Module**

The task of the display module is to present the composed panoramic video frames to the viewer. Commonly, we use CRT monitors or LCD monitors as the main display units. However, they have the problem very limited field of view. Using such display devices cannot obtain the benefit of the panoramic video. So, instead of using the traditional device, immersive display is employed in our system.

A variety of immersive displays have been introduced in recent years

[24, 25, 26, 27, 28, 29], initially in academic institutions and eventually as commercial products. The typical immersive display includes multiple projects situated for front or rear projection onto a large screen. Screen geometry can be flat, curved, or of any irregular shape, while the materials have included solid single-piece walls, flexible screens and tiled panels. Currently our system is using the VisionStation from Elumens[26] (Figure 3.5). It is a pseudo- hemispherical display that allow user to be immersed in the 3D digital content. This display is most suitable for displaying our panoramic video.



Figure 3.5: VisionStation

## 3.2 Design Issues

In designing the system architecture, a number of considerations have been taken into account for performance, scalability, and reliability.

### 3.2.1 Modular Design

The framework is designed with a number of modules. The modules communicate with each others with simple data interfaces. Updating a module can include new functionality to the system. For example, video capture module can be replaced by a module that read video data from disk storages, or the

display module can be changed to a conventional monitor for displaying the live generated panoramic video.

### **3.2.2 Scalability**

Besides the scalability of the modular design, scalability is allowed within each module.

The more the number of cameras, the larger the field of view of the generated panoramic images. To obtain a complete spherical panorama, one has to use enough cameras. Our approach does not limit the number of camera used. Instead of using more cameras, the cameras may be changed to those with larger field of view which will also result in the generation of larger field of view panoramic images.

Since the network configuration varies very much in different networks. In designing the streaming module, instead of fixed using a specific video coding scheme and the transport protocol, containers of video coding scheme and a container of network transport protocol are designed. The video coding container supports a variety of video codecs, such as MPEG, MJPEG, H.261, Indeo and etc. The transport protocol containers supports UDP/IP, TCP/IP and RTP. With different network properties, one most suitable combination can be employed in order to achieve the highest performance.

### **3.2.3 Workload distribution**

In designing a real-time application, the time critical characteristic has to be considered. For our live video application framework, the amount of time between each video frame is very short of about 30 milliseconds. The amount of processing can be done is very small. Therefore, in our design, we put all processing steps that are computation intensive in an offline step.

Precomputation is done to minimize the amount of work to be done in each video frame cycle. In the running time, two main tasks are carried out which cannot be done in advance in the preprocessing phase. One of them is real-time compression of the live captured video stream in the capture unit. Using sophisticated compression codec, it is possible to compress in real-time. The receiver has to do the decompression of the video data received and then do the real-time rendering of the wide field of view panoramic video.

## Chapter 4

# Panoramic Video Mosaic

In this chapter, we will discuss how the separated live video streams can be stitched together in real-time to produce a live panoramic video.

### 4.1 Video Mosaic to Image Mosaic

Image mosaicing is the task of combining a collection of images with small fields of view to obtain a larger field of view image. Video Mosaic, as a term used by other researchers[20, 17, 21], usually means the construction of a large field of view image from a single video stream. Instead of using this definition, we give the term another meaning: video mosaic is the task of combining a collection of video stream with small fields of view to obtain a larger field of view video. Instead of tackling the problem of producing the video panorama in the first place, we shall first ask a simple question: Is it possible to reduce the video mosaic problem to the easier image mosaic problem? If the answer is positive, then many of the existing image mosaic techniques can be employed to the video mosaic problem.

### 4.1.1 Assumptions

In our panoramic video mosaic approach, we have made the following assumptions:

1. The optical centers of all cameras are at the same location.
2. The cameras' relative positions and orientations are fixed.
3. There are sufficient region of overlapping between the adjacent camera views for registration and alignment.
4. The input video streams are synchronized.

With these assumptions, our system uses simpler concepts and algorithms to achieve our goal. Therefore, it is vital to justify the validity of the assumptions.

Normal cameras cannot be placed at the same location as they physically occupy certain amount of space. So, the camera optical centers cannot be exactly the same point. Thus, the first assumption is normally invalid unless using some special cameras[3, 36, 4] such as those shown in Figure 4.1.



(a) Dodeca 1000 Immersive Video Camera by Immersive Media Company



(b) wide-field-of-view camera cluster by Majumder and Meenakshisundaram[36]

Figure 4.1: Specially designed camera cluster

Although we do not use those special cameras, that assumption can still be justified under some restriction. As our system targets at viewing outdoor environment, most of the objects in the scene would at a far distance away. If the parallax of any point between any two cameras is less than half a pixel, then it is in effect the same case as the common optical center. To satisfy the assumption, all objects has to satisfy the following equation:

$$D > \frac{2D_e}{A_e} \quad (4.1)$$

where  $D_e$  is the maximum distance among the optical centers of those cameras and  $A_e$  is the maximum angle subtended between 2 pixels in the image planes. Please refer to Appendix A for details.

Normally, relative positions and orientations of the cameras do not change unless explicitly interposed. To avoid any potential turbulence such as perturbation due wind, the cameras can be mount firmly on a stable base. Moreover, as the camera is manually placed in the setup place, it is always possible to arrange the cameras in such a way that there are some overlapping part between adjacent cameras. Thus the second and the third assumptions are guaranteed.

For the last assumption, the input video are assumed to be synchronized. This synchronization can be enforced by the streaming protocol as will be mentioned in Chapter 6.

### 4.1.2 Processing Pipeline

When generating image mosaic, the input is a set of overlapping images. The case of generating video mosaic is similar, the input is a set of overlapping video streams. As video streams are basically consist of sequences of image frames, we can apply the image mosaic techniques to each video

frames. Combining all the output panoramic images in sequence, we can get the panoramic video mosaics. This, however, does not work well as it has done extensive redundant work. Most of the image mosaic algorithms may require a feature detection step, either done automatically or manually. Variation of the detected features in different time instances of the same input video streams may produce different mosaics. Those mosaics, when viewed individually, will not produce any visual artifacts. But when they are played in sequence, the video will be jerky. Moreover, computation requirement of doing image mosaic is not small. Using high-end machines may still require tens of seconds. Doing image mosaic frame by frame is too computational expensive. This maybe possible for offline video production, but it is impractical for real-time, live video applications.

Instead of running the image mosaic algorithm many times, we want to run the algorithm once and apply the result on all the following frames. In this way, the image mosaicing steps can be done in the offline phase. During the run time, the mosaicing result is used for the all frames and this saves the expensive computation. Base on this idea, the processing pipeline is designed as shown in Figure 4.2. The algorithm can be divided in two phases. In the first offline phase, all the necessary pre-computations are done here. In the online phase, minimal work is carried out to create and display the panoramic video stream.

In the online phase, very little time (of the order of 10 milliseconds) is available for doing computation. Therefore, in the offline phase, it has to do all the steps that can be pre-computed so as to reduce the workload in the real-time online processing phase. The offline phase includes the following steps:

1. *Calibration of cameras*

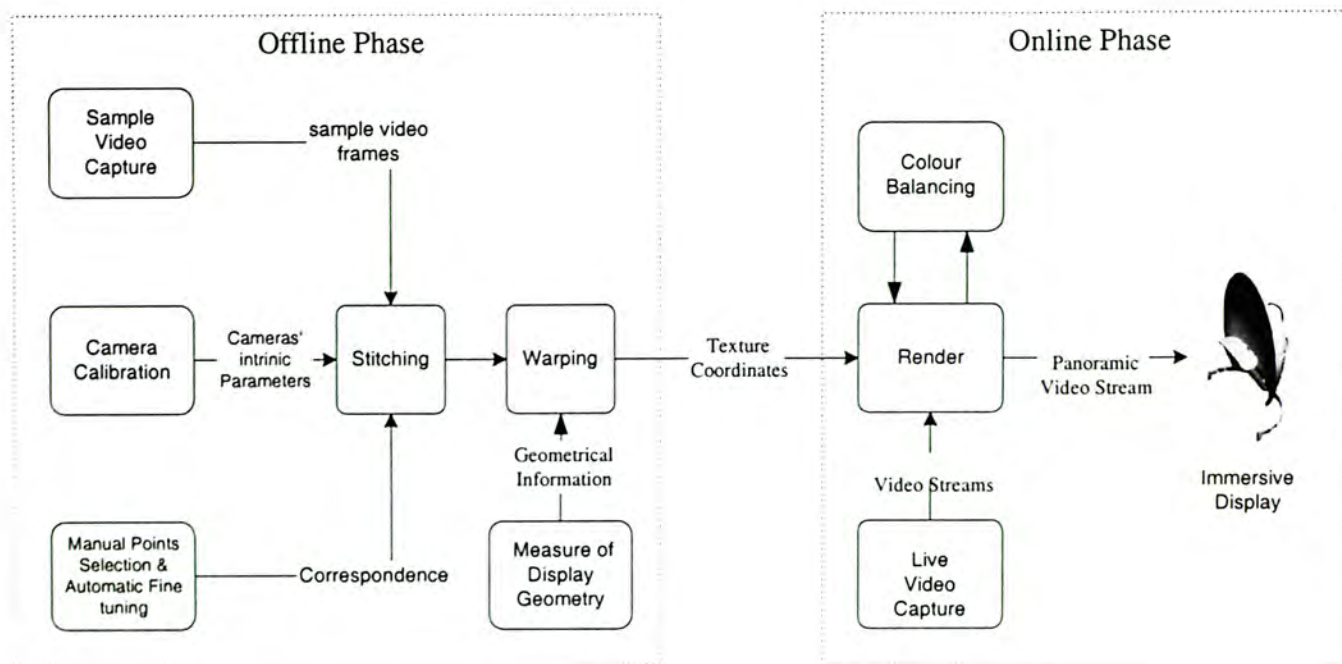


Figure 4.2: Processing Pipeline

Each of the cameras is calibrated separately for its intrinsic parameters and the lens distortion parameters.

## 2. Finding Correspondence

Corresponding points are manually identified for the computation of mosaic. Automatic fine tuning is done to obtain a better correspondence match.

## 3. Mosaic Computation

Using the image mosaic technique as will be addressed in later sections, an initial mosaic is computed. The computed mosaic is triangulated and the texture coordinates of the vertices are retrieved.

In the online phase, the task is the real-time rendering of the panoramic video. Upon receiving all video frames of the same instance, they are used as the texture images. With the computed triangulation and the texture coordinates, the separate frames are blended together to form a large field of view image. Since most conventional graphics accelerators have the support

of the drawing of texture mapped triangles, this rendering can be done in real-time.

## 4.2 Camera Calibration

Before doing our video mosaic, the first task is to acquire live video streams from the surroundings. This task can be achieved by using some video cameras such as charge coupled devices (CCD). There are many types of video cameras, ranging from analog to digital, variable focal length and exposure. Though there are many different types, they can be modeled with a camera model. The ideal pin-hole camera model, with the introduction of distortion correction, can model physical cameras reasonably well as a perspective projection.

### 4.2.1 Perspective Projection

The pinhole camera model shown in Figure 4.3 is adopted in our system. Using this camera model, a camera can be considered as a simple perspective projection. The center of projection (COP) is the camera's optical center and  $f$  is the focal length.

Figure 4.4 shows the process of image formation by the perspective projection where a point  $\mathbf{x}_s = (x, y, z)$  is viewed by a camera at the position  $\mathbf{x}_p = (x_p, y_p, z_p)$ . The rotation from the global coordinate system  $\mathbb{R}^3$  to the camera coordinate system  $\mathbb{R}_c^3$  is specified by a  $3 \times 3$  rotation matrix  $\mathbf{R}$ . Transforming the coordinate system, the point  $\mathbf{x}_s$  becomes  $\mathbf{x}_c = (x_c, y_c, z_c)$  in camera space:

$$\mathbf{x}_c = \mathbf{R}(\mathbf{x} - \mathbf{x}_p) \quad (4.2)$$

Then, the perspective projection scales the x- and y-coordinates by depth

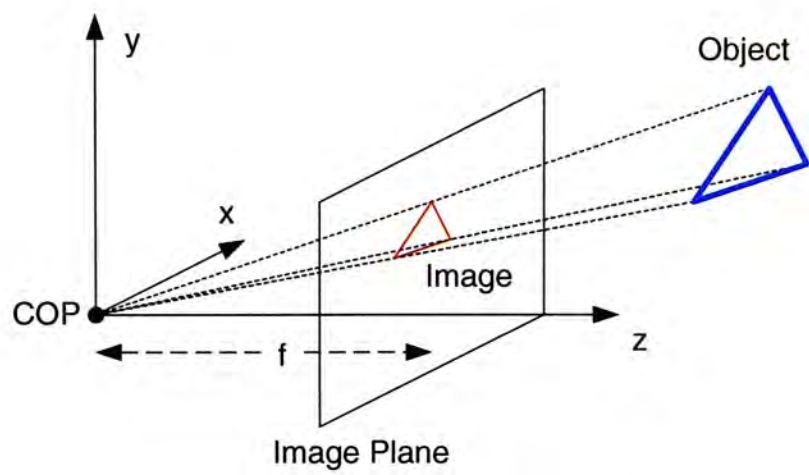


Figure 4.3: Pinhole Camera Model

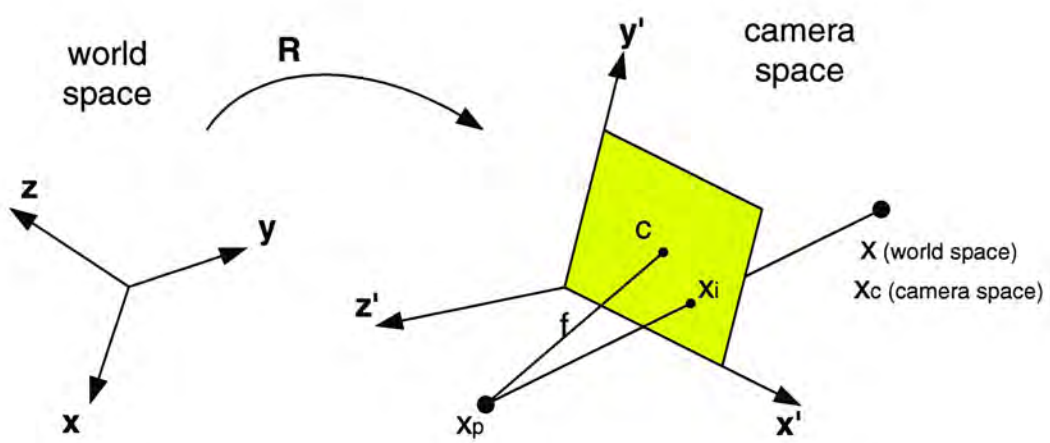


Figure 4.4: Illustration of Perspective Projection

to yield the normalized imaged coordinates  $\mathbf{x}_n(x_n, y_n)$ :

$$x_n = \frac{x_c}{z_c}, \quad y_n = \frac{y_c}{z_c} \quad (4.3)$$

The normalized image coordinates are converted to pixel coordinate position based on the focal lengths  $\mathbf{f}(f_u, f_v)$  and the coordinates  $\mathbf{c}(c_u, c_v)$  of the principal point:

$$\begin{aligned} x_i &= f_u x_n + c_u \\ y_i &= f_v y_n + c_v \end{aligned} \quad (4.4)$$

Note that in the above equation, we have made the assumption that pixels are rectangular. This assumption is valid for most cameras in the market and it is valid for the camera used in our system. In case of non rectangular pixels, one may introduce the skew factor into the model easily[37]. For simplicity, we ignore the skew factor.

The image center and the principal point do not coincide in general, but they have been offset by a few pixels. It is convenient to represent the entire transformation sequence as a chain of matrix transformation, using the geometry in the projective space:

$$\begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} \simeq \mathbf{KM} \begin{pmatrix} \mathbf{R} & -\mathbf{R}\mathbf{x}_p \\ \mathbf{0} & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \quad (4.5)$$

where  $\mathbf{K}$  is a  $3 \times 3$  matrix representing the intrinsic parameters of the camera:

$$\mathbf{K} = \begin{pmatrix} f_u & 0 & c_u \\ 0 & f_v & c_v \\ 0 & 0 & 1 \end{pmatrix} \quad (4.6)$$

and  $\mathbf{M}$  is a  $3 \times 4$  matrix for reducing the dimensionality of the matrices from 3D to 2D:

$$\mathbf{M} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad (4.7)$$

The sign " $\simeq$ " used here means the equality is true up to a non-zero scaling factor. This is because all points which is in front of the cameras and are collinear with the line  $x_p x_c$  will be mapped onto the same point  $x_i$  (Figure 4.4).

### 4.2.2 Distortion

Real cameras usually deviate from the camera model describe before. Distortions in lenses can be decomposed into three components: shift of the optical center, radial distortion and de-centering distortion. They exhibit barrel distortion, pincushion distortion or any other irregular distortion (Figure 4.5). This distortion is more significant when the field of view of the camera is large. Although it may not be apparent when viewing a single image, the distortion may cause noticeable mismatch when stitching images together.

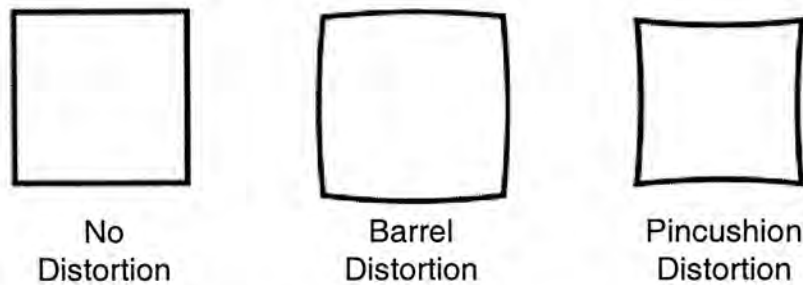


Figure 4.5: Different types of lens distortion

Radial distortion usually dominates the overall distortion function as reported in different literatures[38, 39, 37]. This radial distortion can be modeled by the Equation 4.8 which relates the normalized distorted  $\mathbf{X}_d(x_d, y_d)$

and distortion-free  $\mathbf{X}(x, y)$  image coordinates.

$$\mathbf{X}_d = (1 + k_0 r^2 + k_1 r^4 + k_4 r^6) \mathbf{X} + \mathbf{d}_X \quad (4.8)$$

where  $r^2 = x^2 + y^2$ ,  $k_0$  to  $k_4$  are the distortion coefficients and  $\mathbf{d}_X$  accounts for the tangential distortion:

$$\mathbf{d}_X = \begin{bmatrix} 2k_2xy + k_3(r^2 + 2x^2) \\ k_2(r^2 + 2y^2) + 2k_3xy \end{bmatrix}$$

The inverse problem of computing the normalized image coordinate  $(x, y)$  from the pixel coordinate  $(x_d, y_d)$  is important as our input is the distorted image. However, this inverse mapping is not as simple as the direct pixel mapping because there exists no analytic equation for the inverse. To calculate the inverse, an iterative numerical approximation method is used. It is known that the distorted point will not deviate very much from the distortion-free pixel location. We choose the  $(x_d, y_d)$  as the initial guess and we can get  $(x, y)$  in a few iterations.

### 4.2.3 Calibration Procedure

To determine the internal geometric and optical characteristics of the cameras, we employ the camera calibration toolbox of Open Source Computer Vision Library (OpenCV). This toolbox is based on a number of calibration methods such as Zhang[38], Heikkila [40], Sturm and Maybank[37]. The procedure requires the camera to capture a planar pattern shown at  $\mathbf{N}$  different orientations as shown in Figure 4.6.

In the first step, a linear parameter estimation by the direction linear transformation (DLT) is applied to find the closed-form solution of the calibration problem. In this step, lens distortion cannot be incorporated and

therefore, distortion effects are not generally corrected. So, in the second step, a non-linear estimation is applied by minimizing the residual between the distortion model and the  $\mathbf{N}$  observations. The objective function is expressed as a sum of squared residuals:

$$F = \sum_{i=1}^N (U_i - u_i)^2 + \sum_{i=1}^N (V_i - v_i)^2 \quad (4.9)$$

where  $(U_i, V_i)$  is the observed image point and  $(u_i, v_i)$  is the correct image point determined in the first step.

Perspective projection is generally not a shape preserving transformation. So, in the third step, a correction for asymmetric projection is applied. And in the final step, it solves the image correction problem. Image correction is performed by using an implicit model that interpolates the correct image points based on the physical parameters derived in previous steps. After the whole intrinsic parameters calibration procedure, the effective focal lengths,  $f_u$  and  $f_v$  and the principal point  $(c_u, c_v)$  are obtained.

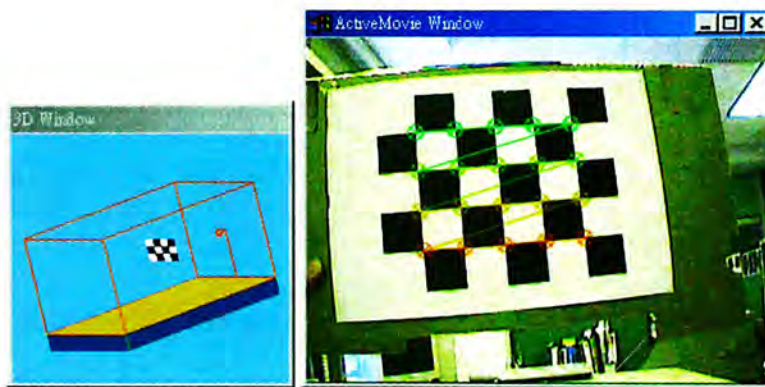
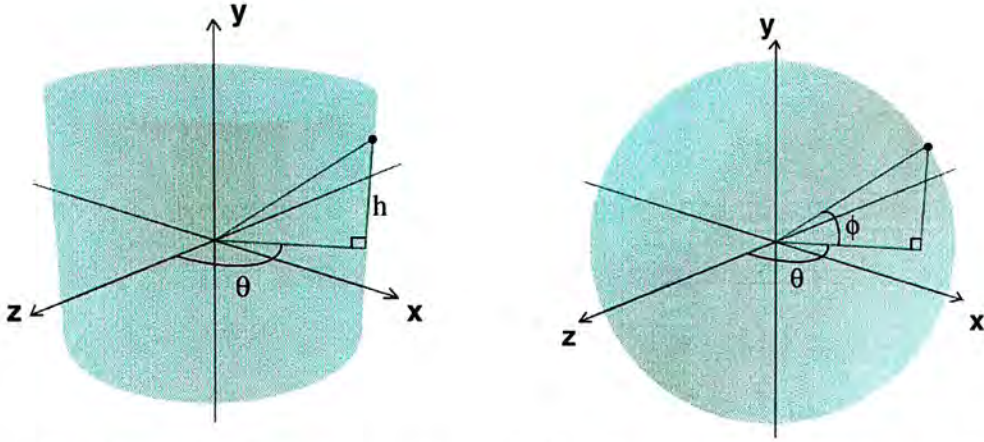


Figure 4.6: Calibration Board

coordinates  $(\theta, \phi)$ . The transformation equation is as follow:

$$\theta = \tan^{-1} \left( \frac{x}{z} \right) \quad (4.12)$$

$$\phi = \tan^{-1} \left( \frac{y}{\sqrt{x^2 + z^2}} \right) \quad (4.13)$$



(a) cylindrical coordinates system (b) spherical coordinates system

Figure 4.7: Common reference coordinate system

After the input images are transformed into either the cylindrical or spherical coordinates, constructing the panorama becomes a pure translation in the new coordinate system. This translation  $(t_x, t_y)$  can be recovered by a minimization process with the error function  $E(t_x, t_y)$ :

$$E(t_x, t_y) = \sum_i [I_2(x_i + t_x, y_i + t_y) - I_1(x_i, y_i)]^2 \quad (4.14)$$

where  $I_j(x, y)$  represents the intensity of the point  $(x, y)$  in image  $j$ . When the translation between two images is found, they are ready to be fused together. To reduce discontinuities in intensity and color between the images being combined, a composite function can be applied to the pixels according to the distance between the edge of the images. Combining more than two images is similar. We first find a reference image, for each other images, it is combined with that reference image in turn using the above two-image method. With

this method, it is possible to stitch reasonably well with synthetic images. However, it has several limitations. This method can only handle the simple case of camera panning. There are also singular points where the method fails. The other limitation is that it requires the focal length to be known in advance. While this method is suitable for creating panorama, it needs a good initial estimation for the relationship between input images.

### 4.3.2 Homography

Instead of transforming to cylindrical or spherical coordinates systems, we use projective transformation as our basis for image mosaicing. As mentioned before, camera can be considered as a perspective projection which is a  $4 \times 3$  matrix. We have the assumption that camera optical centers are overlapping, so depth effect do not occur across two images. The perspective projection equation (4.5) simplifies to:

$$\mathbf{x}' \simeq \mathbf{K}\mathbf{R}\mathbf{x} \quad (4.15)$$

where  $\mathbf{x}' = (x', y', 1)$  and  $\mathbf{x} = (x, y, z)$  Inverting the equation yields:

$$\mathbf{x} \simeq \mathbf{R}^{-1}\mathbf{K}^{-1}\mathbf{x}' \quad (4.16)$$

The equation means that a pixel position in one image is converted into a 3D line in the camera space. Thus, the pixel position in the other image can be obtained by using equation 4.15 again, projecting that 3D lines onto the second image:

$$\mathbf{x}_2 \simeq \mathbf{K}_2\mathbf{R}_2\mathbf{R}_1^{-1}\mathbf{K}_1^{-1}\mathbf{x}_1 \quad (4.17)$$

The 2D projective transformation, or homography, that maps pixel  $(x_1, y_1)$

of image 1 to pixel  $(x_2, y_2)$  of image 2 is:

$$\mathbf{H} = \begin{pmatrix} m_0 & m_1 & m_2 \\ m_3 & m_4 & m_5 \\ m_6 & m_7 & m_8 \end{pmatrix} \simeq \mathbf{K}_2 \mathbf{R}_2 \mathbf{R}_1^{-1} \mathbf{K}_1^{-1} \quad (4.18)$$

This  $3 \times 3$  homography matrix  $\mathbf{H}$  has nine unknowns but of eight degree of freedom. This is due to the fact that there is a scale factor defined in the projective equality. This unknown scale factor can be determined by adding one constraint. Typically, this is done by setting either  $\det(\mathbf{H}) = 1$  or  $m_8 = 1$ . In our case, we have chosen the later one.

### 4.3.3 Homography Computation

Instead of using matrix representation, the homography can be rewritten to the following form ( $m_8$  is set to 1):

$$\begin{aligned} x' &= \frac{m_0x + m_1y + m_2}{m_6x + m_7y + 1} \\ y' &= \frac{m_3x + m_4y + m_5}{m_6x + m_7y + 1} \end{aligned} \quad (4.19)$$

Rearranging the terms, the equation becomes:

$$\begin{aligned} m_0x + m_1y + m_2 - m_6xx' - m_7yx' &= x' \\ m_3x + m_4y + m_5 - m_6xy' - m_7yy' &= y' \end{aligned} \quad (4.20)$$

The eight unknown parameters can be solved without any 3D information using only correspondences of image points. These correspondences can be obtained by feature based methods, or entered by user manually. For each corresponding point pairs, two equations can be setup. A minimum of four pairs have to be identified. In practice, more pairs (around eight) are identified to minimize the corresponding error. With  $n$  points, a system of

$2n$  equations can be formed:

$$\begin{pmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1x'_1 & -y_1x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1y'_1 & -y_1y'_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2x'_2 & -y_2x'_2 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -x_2y'_2 & -y_2y'_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n & y_n & 1 & 0 & 0 & 0 & -x_nx'_n & -y_nx'_n \\ 0 & 0 & 0 & x_n & y_n & 1 & -x_ny'_n & -y_ny'_n \end{pmatrix} \begin{pmatrix} m_0 \\ m_1 \\ m_2 \\ m_3 \\ m_4 \\ m_5 \\ m_6 \\ m_7 \end{pmatrix} = \begin{pmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ \vdots \\ x'_n \\ y'_n \end{pmatrix} \quad (4.21)$$

Solving the system of equations for  $m_i$ , the homography can be constructed. When the number of points is larger than four, the system is over-determined and we wish to find a solution which introduces the least error. In our implementation, we employ the singular value decomposition (SVD) method as our equation solver to find the least-square solution. SVD is based on the following theorem of linear algebra: Any  $M \times N$  matrix  $\mathbf{A}$  whose number of rows  $M$  is greater than or equal to its number of columns  $N$ , can be written as the product of an  $M \times N$  column-orthogonal matrix  $\mathbf{U}$ , an  $N \times N$  diagonal matrix  $\mathbf{W}$  with positive or zero elements, and the transpose of an  $N \times N$  orthogonal matrix  $\mathbf{V}$ . This is illustrated by the following equation:

$$\begin{pmatrix} A \end{pmatrix} = \begin{pmatrix} U \end{pmatrix} \begin{pmatrix} w_1 & & \\ & \ddots & \\ & & w_N \end{pmatrix} \begin{pmatrix} V^T \end{pmatrix} \quad (4.22)$$

In case where the system is singular or numerically very close to singular, this method is superior than techniques like Gaussian elimination or LU decomposition. Using the SVD methods,  $m_i$  can be determined by the following equation:

$$\begin{pmatrix} m_0 \\ \vdots \\ m_7 \end{pmatrix} = \begin{pmatrix} & \\ & V \\ & \end{pmatrix} \begin{pmatrix} & \\ & diag(1/w_i) \\ & \end{pmatrix} \begin{pmatrix} & \\ & U^T \\ & \end{pmatrix} \begin{pmatrix} x'_1 \\ y'_1 \\ \vdots \\ x'_n \\ y'_n \end{pmatrix} \quad (4.23)$$

Using the homography matrix obtained by this method, images can usually be registered together reasonably well, provided that the selected points are correct. However, this method depends heavily on the selected points. With some errors in the correspondence, the homography found will not register the images correctly. So, a fine adjustment step has to be carried out to correct the homography matrix in case there are any mis-registration.

#### 4.3.4 Error Minimization

Instead of using selected points to find the homography matrix, the fine adjustment step use all the two input images. This step minimizes the sum of the squared intensity errors

$$E = \sum_p [I'(x', y') - I(x, y)]^2 = \sum_p e_p^2 \quad (4.24)$$

over all the corresponding pair of pixels  $p$  which are inside both image  $I(x, y)$  and  $I'(x', y')$ . With this minimization, the previous approximated homography can be further improved.

To perform the minimization, we use the standard non-linear least squares minimization - the Levenberg-Marquardt method[18, 9]. This method is a combination of the steepest gradient decent and the Newton's methods for minimization. It constructs good search directions and distances to move based on the target function near the initially guessed minimum, and then progressively moving towards the dominant local minimum.

The optimization process requires the determination of the derivatives of each error term  $e_p^2$  with respect to  $\mathbf{H}$ . The derivative  $\frac{\partial e_p^2}{\partial \mathbf{H}}$  is expressed as an eight component vector consisting of derivatives with respect to each of the  $m_i$  in  $\mathbf{H}$ :

$$\frac{\partial e_p}{\partial m_i} = \frac{\partial I'}{\partial x'} \frac{\partial x'}{\partial m_i} + \frac{\partial I'}{\partial y'} \frac{\partial y'}{\partial m_i} \quad (4.25)$$

Or explicitly:

$$\begin{aligned} \frac{\partial e_p}{\partial m_0} &= \frac{x}{\mathbf{D}_p} \frac{\partial I'}{\partial x'} & \frac{\partial e_p}{\partial m_1} &= \frac{y}{\mathbf{D}_p} \frac{\partial I'}{\partial x'} \\ \frac{\partial e_p}{\partial m_2} &= \frac{1}{\mathbf{D}_p} \frac{\partial I'}{\partial x'} & \frac{\partial e_p}{\partial m_3} &= \frac{x}{\mathbf{D}_p} \frac{\partial I'}{\partial y'} \\ \frac{\partial e_p}{\partial m_4} &= \frac{y}{\mathbf{D}_p} \frac{\partial I'}{\partial y'} & \frac{\partial e_p}{\partial m_5} &= \frac{1}{\mathbf{D}_p} \frac{\partial I'}{\partial y'} \\ \frac{\partial e_p}{\partial m_6} &= -\frac{x'}{\mathbf{D}_p} \left( \frac{\partial I'}{\partial x'} + \frac{\partial I'}{\partial y'} \right) & \frac{\partial e_p}{\partial m_7} &= -\frac{y'}{\mathbf{D}_p} \left( \frac{\partial I'}{\partial x'} + \frac{\partial I'}{\partial y'} \right) \end{aligned} \quad (4.26)$$

where  $\mathbf{D}_p = m_6x + m_7y + 1$ .

The overall gradient term  $\mathbf{g} = (g_0, g_1, \dots, g_7)$  is computed by accumulating over all the error terms:

$$g_i = -2 \sum_p e_p \frac{\partial e_p}{\partial m_i} \quad (4.27)$$

The Levenberg-Marquardt method requires the calculation of a second derivative matrix (Hessian matrix)  $\mathbf{Q}$ . However, there is no way of directly evaluating the Hessian matrix. We can only evaluate the function to be minimized and its gradient. Therefore, we build an approximation of the

Hessian matrix  $\mathbf{Q}$  with the components as follow:

$$q_{ij} = \sum_p \frac{\partial e_p}{\partial m_i} \frac{\partial e_p}{\partial m_j} \quad (4.28)$$

The homography vector  $\mathbf{H}$  is then updated by

$$\Delta \mathbf{H} = -(\mathbf{Q} + \lambda \mathbf{I})^{-1} \mathbf{g} \quad (4.29)$$

where  $\mathbf{I}$  is the identity matrix, and  $\lambda$  is a stabilization parameter set initially to a high value, and gradually reduced to zero as the optimization converges.

We use the homography matrix determined by SVD as the initial guess for the minimization process, the recipe is as follows:

1. Compute  $E(\mathbf{H})$ .
2. Pick a modest value for  $\lambda$ , say  $\lambda = 0.001$
3. Solve the equation 4.29 and evaluate  $E(\mathbf{H} + \Delta \mathbf{H})$ .
4. If  $E(\mathbf{H} + \Delta \mathbf{H}) \geq E(\mathbf{H})$ ,  $\lambda$  is increased by a factor of  $t$ , say  $t = 10$  and go back to step 3.
5. If  $E(\mathbf{H} + \Delta \mathbf{H}) \leq E(\mathbf{H})$ ,  $\lambda$  is decreased by a factor of  $t$ , and the homography is updated :  $\mathbf{H} \leftarrow \mathbf{H} + \Delta \mathbf{H}$  and go back to step 3.
6. Iteration stops until the error function evaluates to a value below a threshold or a fixed number of steps has been iterated.

### 4.3.5 Stitching Multiple Images

The discussion in previous sections considers only the case of fusing two input images. For  $N$  input images, the same method as fusing two images is used. In the first round, adjacent image pairs are combined together, resulting in

$\frac{N}{2}$  images. Recursively, the adjacent image pairs from that newly formed  $\frac{N}{2}$  images are combined to form  $\frac{N}{4}$  images. This is done repeatedly until one single large field of view image is formed. This hierarchical approach is shown in Figure 4.8.

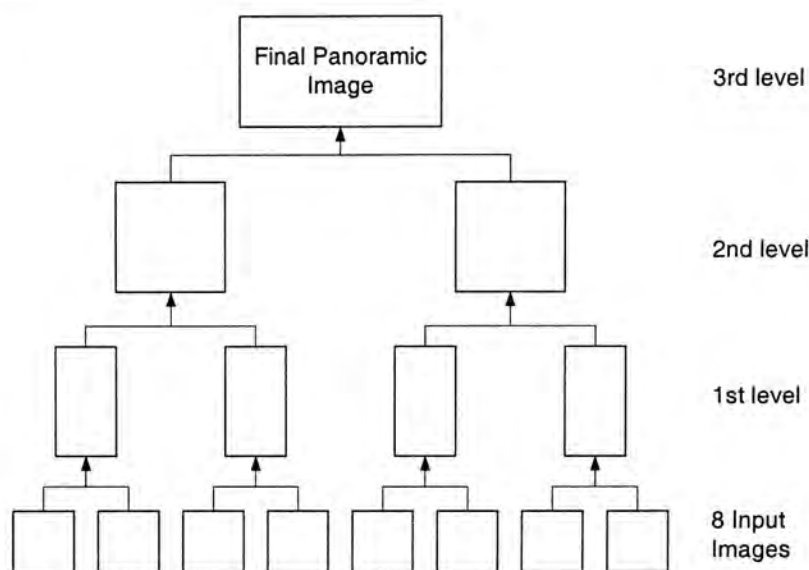


Figure 4.8: Stitching Hierarchy

In each round, two adjacent images is combined together. To find the adjacent image pairs, we use the information from the camera arrangement. As mentioned in Chapter 3, each camera group has four cameras and each two of them are fixed together. In the first round, the image pairs from those cameras which are fixed together are stitched first. Then the two fused image from the same camera group are stitched. In the final step, the fused images from different camera group are stitched together to form a large field of view image.

### 4.3.6 Seamless Composition

After registering all the input images, the images are composed into a complete mosaic. However, there are inevitably errors in the registration process

resulting from incompatible model assumptions, dynamic scenes, etc. Furthermore, in most cases image that need to be combined are not exposed evenly due to the varying lighting conditions in different directions and the build-in automatic gain control in the cameras. So, this undesirable effects should be alleviated in this compositing process.

Typical method to do the blending is to assign weighted blending functions to the source images. There are more sophisticated blending methods such as blending with multiresolution spline[23]. However, they are too complex to be executed in real-time. Instead, a weighting function that decreases near the boundary of a source image is used in our system. Besides the simplicity, this simple weighting function has the advantage that it can be executed very fast with the graphics hardware acceleration using the alpha blending.



Figure 4.9: A mosaic formed by seamlessly combining 7 images

## 4.4 Image Mosaic to Video Mosaic

All the discussion presented so far does not take into account the temporal element of video, assuming that techniques for generating image mosaic can be directly applied to the generation of video mosaic. However, there are problems which are unique to the video mosaic.

### 4.4.1 Varying Intensity

In almost all video cameras, they have a built-in automatic gain control (AGC). This AGC adjusts the captured image depending on the lighting condition. If the scene is dark, AGC will try to brighten the scene and if the scene is bright, AGC will try to darken the scene. This can ensure that the captured image will be in suitable lighting condition. AGC does not change continuously, in contrast, it changes rather abruptly. When the lighting condition of the surrounding falls below a lower limit or raise above an upper limit, it will move in a great step towards a normal intensity level. In the case of one camera, intensity of the whole image changes. For our  $n$ -cameras system, each of the  $n$  cameras has its own lower and upper limit. So, when the lighting condition changes, some may change the intensity level inconsistently, resulting in mismatch of the color intensity across images when they are combined together.

Using cameras with fixed exposure can solve the problem caused by the AGC. But using this kind of cameras has other limitations. They are usually old-fashioned, obsolete and of poor quality (both image resolution and color). In addition, they have a very limited intensity range which cannot operate in dim environment. Therefore, using camera with AGC still is a better choice unless high-end controllable exposure camera is available.

There is little thing that can be done on the problem of varying intensity

as how the AGC works and its parameters are usually unknown. To deal with that, one heuristic approach is to periodically adjust the intensity globally among different video streams. This can help improving the video quality by a certain extent.

### 4.4.2 Video Frame Management

As our video streams are obtained through the network, there are many network related problems that may affect the obtained video frames. One important aspect is synchronization. Our stitching method has made the assumption that the input video streams are synchronized. If the input images used in the calibration step are not of the same time instance, the computed relationship between the images may not be valid for applying on the subsequent video frames in the running time.

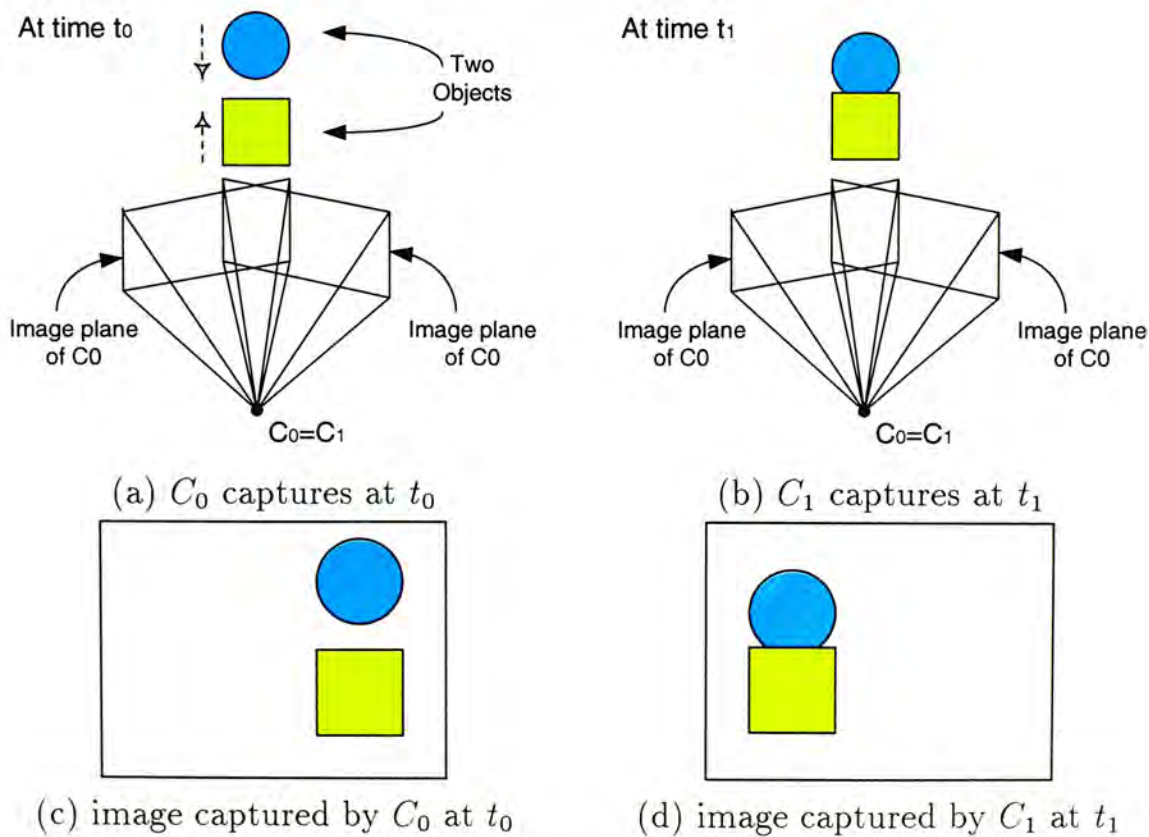


Figure 4.10: Video capture without synchronization

This concept is illustrated in Figure 4.10 a,b. Camera  $C_0$  captures the scene as  $I_0$  at the time  $t_0$  and camera  $C_1$  captures the scene as  $I_1$  at the time  $t_1$ . In that period of  $t_1 - t_0$ , the sphere moves down while the square moves up. It is obvious that using captured image  $I_0$  (Figure 4.10c) and  $I_1$  (Figure 4.10d). to compute the homography will not yield valid result. In the other way, using a correct homography may fail to merge two images taken at different time.

Another problem associated with the network is the loss of video frame. With missing video frames, rather than discarding the other video frames of that particular instance, we use the previous frame as the replacement. This will be the same problem as the unsynchronized video frames before. But this situation is expected to appear only occasionally. On the other hand, this can improve the smoothness of the video produced.

This network related problems will be addressed in detail in Chapter 6.

## Chapter 5

# Immersive Display

As described in Chapter 2, there are a lot of immersive displays available in academic[24, 25] and also in commercial community[27, 26, 29, 28]. All of them try to provide the user the sense of immersion of the virtual environment. To achieve that, the main problem is to fool the users physiologically and psychologically.

### 5.1 Human Perception System

As we want to immerse the user into the computer generated virtual environments, it is necessary to fool the user's senses. The contribution of each of the five human senses[41] is shown in Table 5.1. It shows clearly that human sight provides the major part of information for user to understand the environments. Therefore, visual presentation becomes the central issue in developing our tele-immersive applications.

To give user a visually immersive feeling, it is necessary to provide as much visual information as possible to the user. The ideal case is having the whole virtual scene surrounds the user. The total horizontal field of vision of both human eyes is about  $180^\circ$  without eye movement. If allowing eye movements to the left or right, the total field of vision possible with moving

sense	contribution
sight	70%
hearing	20%
smell	5%
touch	4%
taste	1%

Table 5.1: Contribution of Human Senses

the head is about  $270^\circ$ . The vertical field of vision is typically over  $120^\circ$ . While the total field is not necessary for a user to feel immersed in a virtual environment, it is believed that at least  $110^\circ$  is necessary for the horizontal field of vision.

## 5.2 Creating Virtual Scene

To create the virtual scene for the immersion, the actual steps vary greatly depending on the projector type, screen geometry and the users position. In general, all methods use the same principle. It relocates the *sweet spot* to the desired viewing point.

Sweet spot represents an ideal viewing position where a correct image can be perceived when viewing from that point. By default, this would be the area around the projector. However, this position is normally not available to a viewer due to the presence of physical objects such as the projector or other structures. Once the viewer moves away from the sweet spot, he/she will perceive a distortion of the image. To overcome this problem, the sweet spot should be relocated to the viewer position. This would ensure that the viewer perceives the virtual environment that is distortion-free.

To relocate the sweet spot, it is necessary to know the position  $\mathbf{P}_p$  and  $\mathbf{R}_p$  orientation of the projector, the position of the viewer  $\mathbf{P}_e$  (sweet spot), and the geometry of the display surface. The virtual camera is positioned

at the desired viewer location (sweet spot). Rays are cast from this virtual camera to the display surface. For a point  $\mathbf{u}$  on the camera image plane, it will project onto the point  $\mathbf{P}_s$ . The projector will see that  $\mathbf{P}_s$  as  $\mathbf{v}$  on the projector image plane. Therefore, a relationship can be established between  $\mathbf{u}$  and  $\mathbf{v}$ . This process is shown in Figure 5.1.

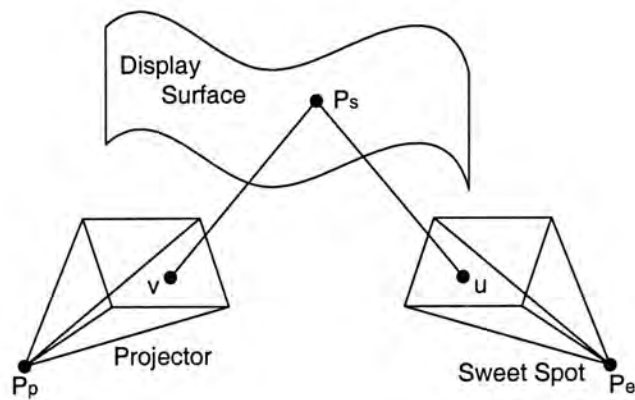


Figure 5.1: Relocation of Sweet Spot

### 5.3 VisionStation

VisionStation (Figure 3.5) is a single projector, Using a patented TruTheta lens (fisheye like lens) that can provides 180 degree field of view, a single projector can fill the hemispherical screen with images. The primary reason to use VisionStation as the display is that it is simple to use, easy to configure, have a known shape and well documented system parameters (such as lens properties). It has numerous advantages over other display systems. The first advantage is the wide field of view. VisionStation has a large hemispherical projection screen which engulfs the viewer, allowing about  $180^\circ$  horizontal field of view and  $135^\circ$  in vertical direction. It can present a larger portion of a visual environment than conventional flat-screen displays.

Most other dome-based systems [42, 43, 30, 44, 45] rely on multiple projectors and seamed together output from multiple computers, making them complicated to configure and expensive. For the VisionStation, the physical hardware consists of one projector and one hemispherical projection screen which is operated by a single computer. It is therefore easier to configure and more portable. Because of its single projector and single projection screen, it does not require computer enhanced edge blending technologies which surely simplify the work to use it as a displaying device.

### 5.3.1 F-Theta Lens

Ordinary projector lens has the property that image height is proportional to the tangent of the angle, i.e. a perspective projection. This is suitable for normal planar display. And it has also the restriction of limited field of view. For a spherical display like VisionStation, however, ordinary lens does not suit well.

VisionStation employs a patented optical system[46], using a special lens system called TruTheta lens (Figure 5.2). This lens enables high-quality projection on a hemispherical screen as the common f-theta lenses but it allows much wider projection angle, that is 180 degree. Also, it can create images that are consistently bright, even and in focus. It distributes pixels equally throughout the hemispherical surface, eliminating any distortion.

In contrast to normal lens which provides perspective projection, f-theta lens has a property that an array of image pixels is projected into a hemispherical projection having constant angular separation among adjacent image pixels. This can be described by the following equation:

$$l = k\theta \tag{5.1}$$

where  $l$  is the length from the image center,  $k$  is a proportionality constant

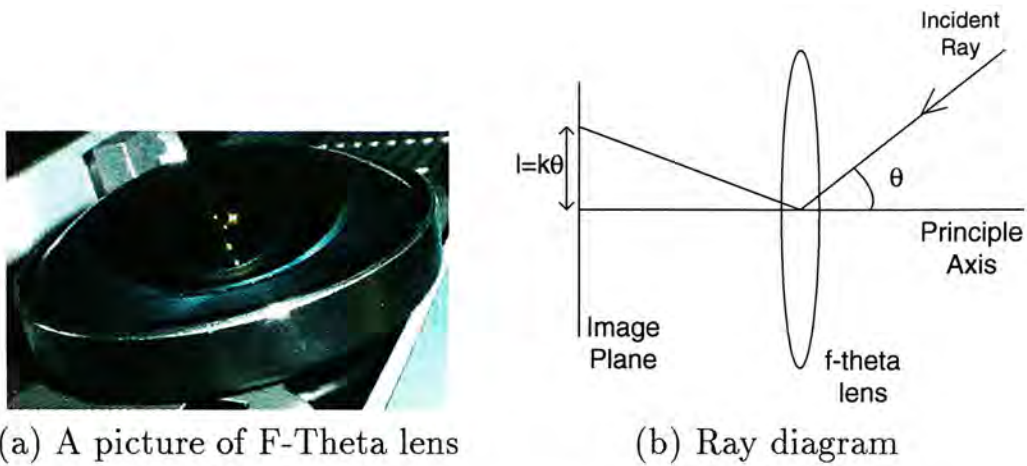


Figure 5.2: F-Theta lens

and  $\theta$  is the deviation angle from the lens principle axis. This type of lenses are used mainly in scanning system like engraving and labeling system, phototypesetting system, etc.

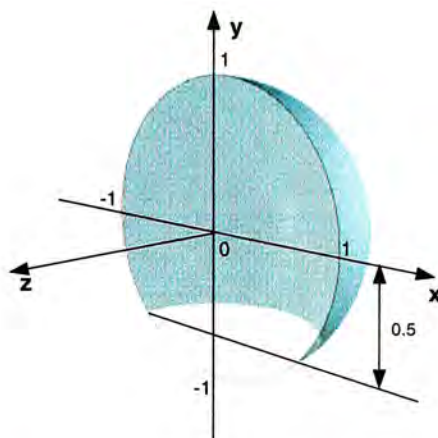
### 5.3.2 VisionStation Geometry

In order to project properly using the VisionStation, it is necessary to understand its geometry. Figure 5.3 depicts the coordinate systems we use on the VisionStation. We use a normalized 3D coordinate system where the origin  $(0,0,0)$  coincides with the center of the hypothetical sphere of VisionStation projection screen.

- The x-axis goes to the right, where  $x = 1$  touches the right edge of the hypothetical hemispherical screen (looking into the screen).
- The y-axis goes up, where  $y = 1$  touches the top edge of the VisionStation screen above the center point.
- The z-axis goes away from the screen. All points on the screen would then have a negative z value.



(a) Model of VisionStation



(b) Geometry of VisionStation

Figure 5.3: VisionStation

It should be noted that the projection screen is not a complete hemisphere. The lower part of the screen is cut because that area is usually blocked by the projector. So it is not meaningful to create that lower. It is a truncated dome shaped that all  $y$  smaller than  $-0.5$  will not be projected.

### 5.3.3 Sweet Spot Relocation and Projection

To display images properly on the VisionStation, basically it has to go through three main steps. In the first step, a ray is cast from the eye position  $P_e$  to the projection screen according to the viewing direction  $\alpha_1$  and  $\alpha_2$ . The ray hits the screen at the point  $P_s$ . The next step is to find the viewing angle from the projector position  $P_p$ . A ray is back-cast to the projector and then the viewing angle is determined. In the final step, the normalized image coordinates  $(x_i, y_i)$  of that viewing direction is found.

#### 1. From eye to screen

When a ray is cast via the direction  $(\alpha_1, \alpha_2)$  from  $P_e = (x_e, y_e, z_e)$ , the ray will hit on the projection screen at  $P_s = (x_s, y_s, z_s)$  where

$$\tan \alpha_1 = \frac{x_s - x_e}{z_e - z_s}$$

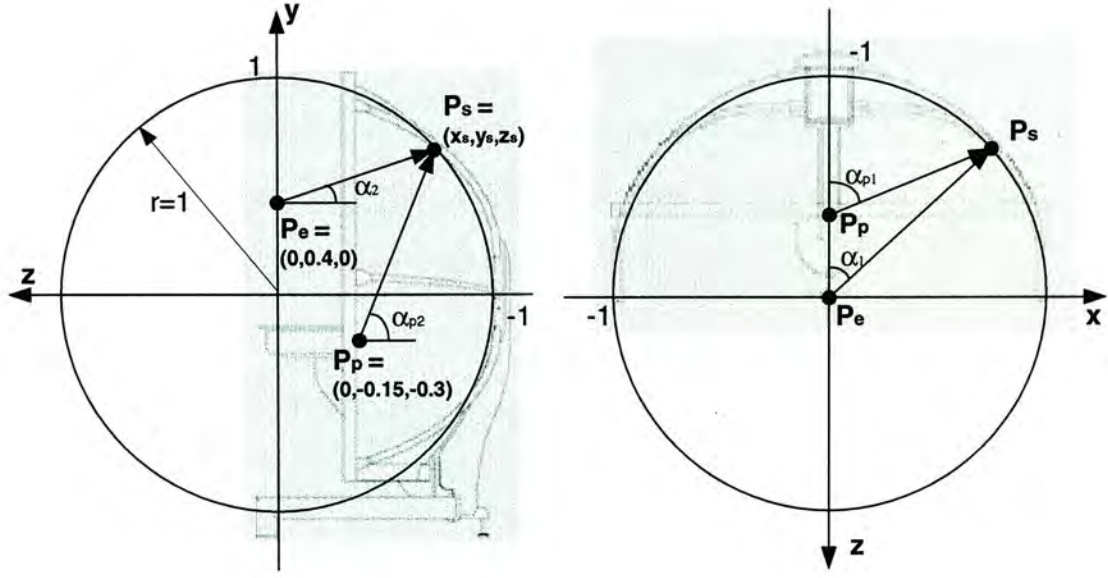


Figure 5.4: Projection on VisionStation

$$\tan \alpha_2 = \frac{y_s - y_e}{z_e - z_s} \quad (5.2)$$

Since the projection screen is of spherical shape, any points on the screen would satisfy the relationship:

$$x_s^2 + y_s^2 + z_s^2 = 1 \quad (5.3)$$

Thus, combining Equations 5.2 and 5.3, we may get a quadratic equation:

$$Az^2 + Bz + C = 0 \quad (5.4)$$

where

$$\begin{aligned} A &= \tan^2 \alpha_1 + \tan^2 \alpha_2 + 1 \\ B &= 2z_e - 2 \tan \alpha_1 x_e - 2 \tan \alpha_2 y_e \\ C &= x_e^2 + y_e^2 + z_e^2 - 1 \\ z_s &= z + z_e \end{aligned}$$

The quadratic equation 5.4 has two solutions. If the solutions are complex, then the ray does not hit on the screen. This case does not

normally appear as the eye position is inside the unit sphere inside the VisionStation. If the solutions are real, the smaller solution  $z_{s1}$  will be the one in front of the eye and the larger solution  $z_{s2}$  will be behind the eye (Figure 5.5). So, the smaller one is the correct solution. With  $z_s$ ,  $x_s$  and  $y_s$  can be computed using Equation 5.2.

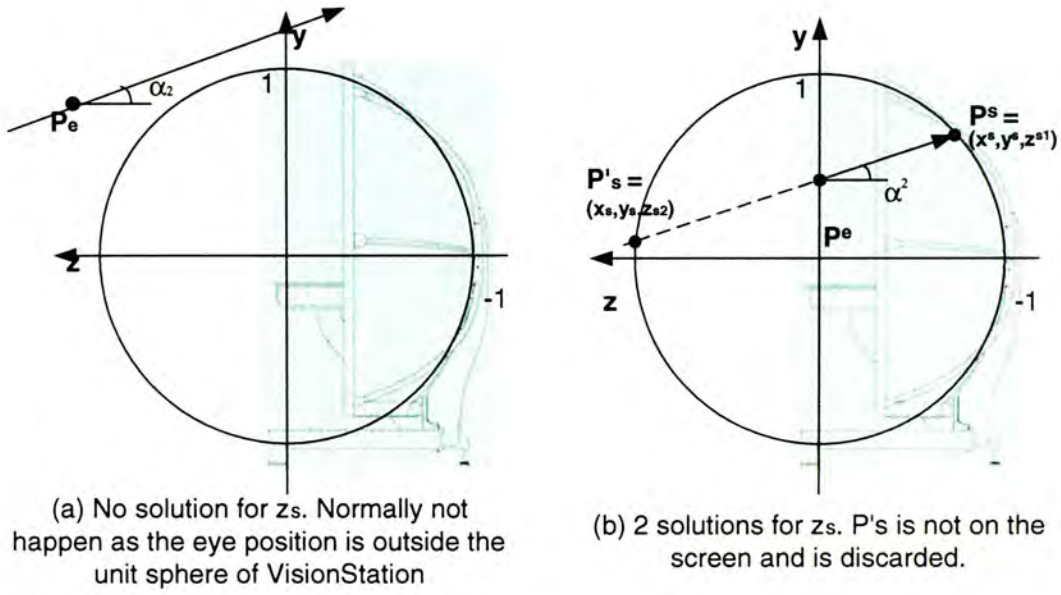


Figure 5.5: Solutions of  $z_s$

## 2. From projection screen to projector

Having found the screen point  $P_s$ , we need to find how the projector will see that particular point. The horizontal and vertical viewing angles  $\alpha_{p1}$  and  $\alpha_{p2}$  from the projector  $P_p = (x_p, y_p, z_p)$  can be found by the following equations:

$$\begin{aligned}\alpha_{p1} &= \tan^{-1} \left( \frac{x_s - x_p}{z_p - z_s} \right) \\ \alpha_{p2} &= \tan^{-1} \left( \frac{y_s - y_p}{z_p - z_s} \right)\end{aligned}\tag{5.5}$$

## 3. From projector to image

Depending on the projector, the method of generating images is different. In normal projector, it preserves a perspective projection. But for

the VisionStation, it uses the special F-theta lens which gives a linear relationship between deviation angle and length from image center.

It is required to change the coordinate system to that shown in Figure 5.6 where the origin is taken as the position of the projector:

$$\begin{aligned}\theta_1 &= \tan^{-1} \left( \frac{\tan \alpha_{p2}}{\tan \alpha_{p1}} \right) \\ \theta_2 &= \tan^{-1} \left( \sqrt{\tan^2 \alpha_{p1} + \tan^2 \alpha_{p2}} \right)\end{aligned}\quad (5.6)$$

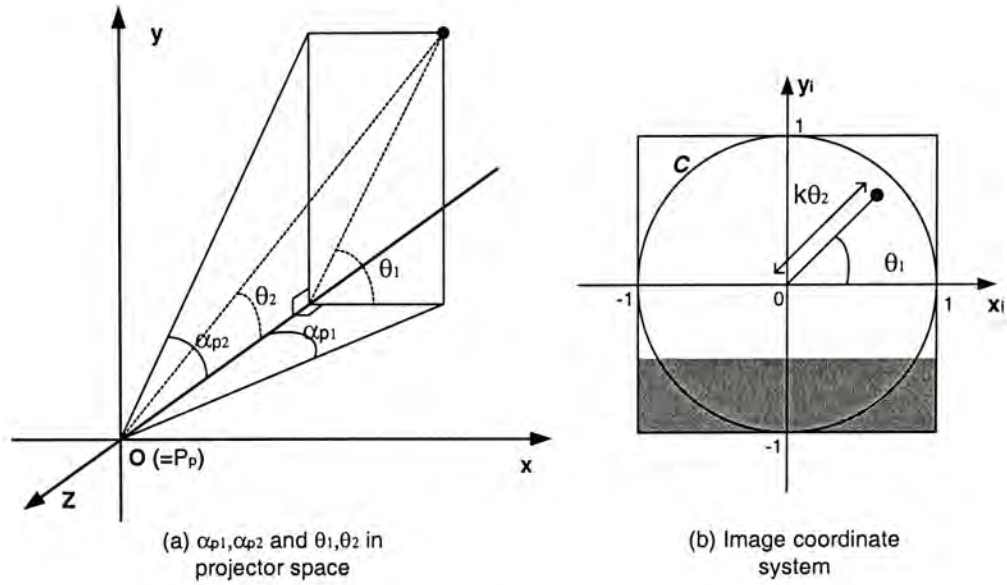


Figure 5.6: Coordinates System of Projector

For each pair of  $\phi_1$  and  $\phi_2$ , it corresponds to a point  $(x_i, y_i)$  on the image which are to be projected. For our image coordinate system, the ranges are  $-1 \leq x_i \leq 1$ , and  $-1 \leq y_i \leq 1$ . It is obvious that when  $\theta_1 = 0$  and  $\theta_2 = \frac{\pi}{2}$ ,  $(x_i, y_i)$  should be  $(1, 0)$ . So according to Equation 5.1,

$$1 = k \left( \frac{\pi}{2} \right)$$

or

$$k = \frac{2}{\pi} \quad (5.7)$$

And thus, the image coordinate  $(x_i, y_i)$  is

$$\begin{aligned} x_i &= \frac{2\theta_2}{\pi} \cos \theta_1 \\ y_i &= \frac{2\theta_2}{\pi} \sin \theta_1 \end{aligned} \quad (5.8)$$

For a full hemispherical screen, all the  $(x_i, y_i)$  inside the circle  $C$  used by the projector to project onto the screen. For VisionStation, the lower part is truncated and will not be seen by the users. All image points with  $x_i < -0.5$  will therefore not be used. In generating the images, all points with  $x_i < -0.5$  will be clipped.

After these three steps, the correct projection is found for the particular eye position and projector position. This is shown at Figure 5.7. Each vertical and horizontal line is separated by 10 degree And the black lines show where  $\alpha_1$  or  $\alpha_2$  being zero.

Because of the special projection system of VisionStation, the projection is not a perspective transformation as in common projector, nor is it a projective transformation. So, common user operations like panning, rotation, or zooming cannot be represented by the simple matrix operations as that used in OpenGL. To achieve those operations, it is necessary to recompute using the described three steps again. This lead to a performance problem when doing such operations.

#### 5.3.4 Sweet Spot Relocation in Vector Representation

Similarly, the sweet spot relocation problem can be solved by using vector representation. Vector mathematics may allow the calculation be done on standard graphics boards. In the first step, a ray is cast from the eye position  $\mathbf{P}_e$  to the projection screen with the unit viewing direction  $\mathbf{V}_e$ . The equation

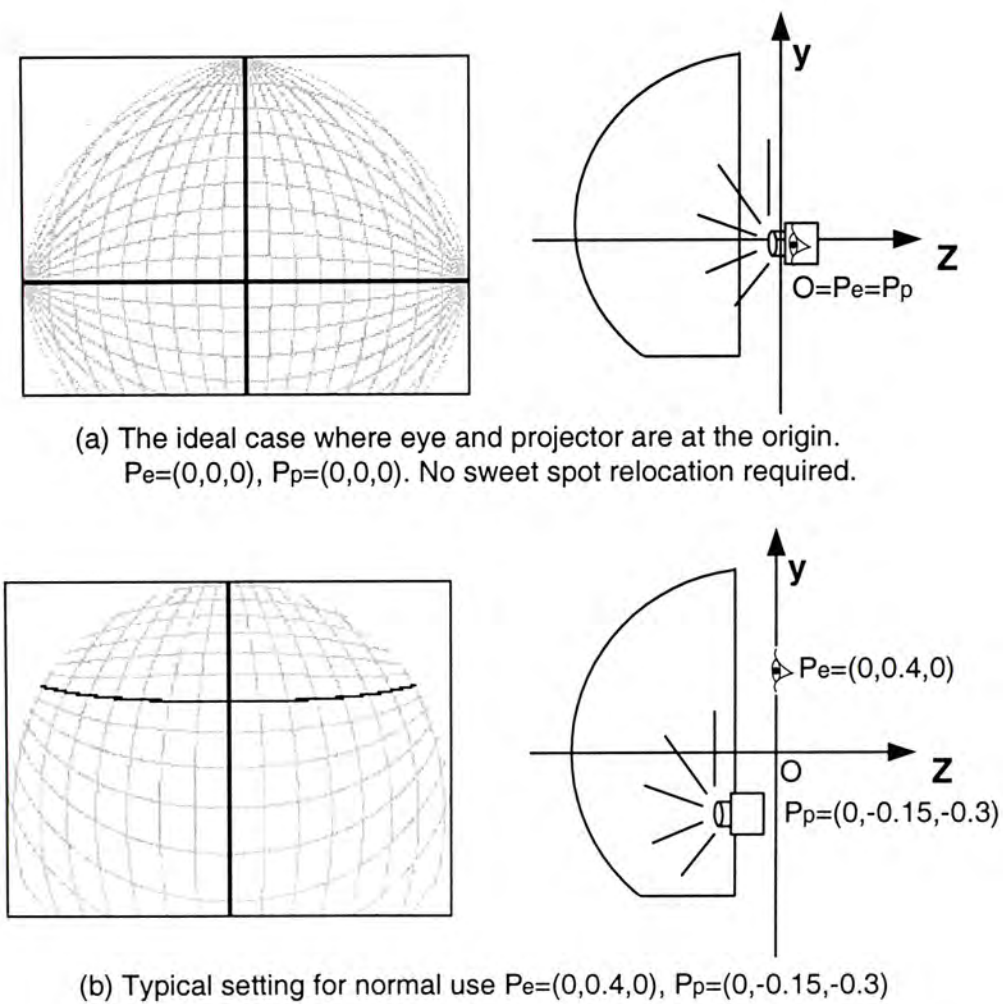


Figure 5.7: Projection for VisionStation

of the ray is therefore:

$$\mathbf{P} = \mathbf{P}_e + t\mathbf{V}_e \quad (5.9)$$

where  $t$  is a scalar number.

The ray hits the projection screen at the point  $\mathbf{P}_s$ , thus:

$$\mathbf{P}_s = \mathbf{P}_e + t\mathbf{V}_e \quad (5.10)$$

Since any point on the screen is the same as on a unit sphere, therefore

$$\begin{aligned} |\mathbf{P}_s| &= 1 \\ |\mathbf{P}_e + t\mathbf{V}_e| &= 1 \\ (\mathbf{P}_e + t\mathbf{V}_e) \cdot (\mathbf{P}_e + t\mathbf{V}_e) &= 1 \\ |\mathbf{V}_e|^2 t^2 + 2\mathbf{P}_e \cdot \mathbf{V}_e t + |\mathbf{P}_e|^2 - 1 &= 0 \end{aligned} \quad (5.11)$$

$t$  can be found by solving Equation 5.11. We discard the value of  $t$  which does not give point  $\mathbf{P}_s$  on the projector screen ( $z_s > 0$ ).

The equation of the ray from the projector to a unit viewing direction  $\mathbf{V}_p$  is as follow:

$$\mathbf{P} = \mathbf{P}_p + s\mathbf{V}_p \quad (5.12)$$

where  $s$  is a scalar number.

Since we want to find a ray that hit the point  $P_s$ , Equation 5.12 becomes:

$$\mathbf{P}_s = \mathbf{P}_p + s\mathbf{V}_p \quad (5.13)$$

Combining with Equation 5.10, we get:

$$\begin{aligned} \mathbf{P}_s &= \mathbf{P}_p + s\mathbf{V}_p = \mathbf{P}_e + t\mathbf{V}_e \\ s\mathbf{V}_p &= \mathbf{P}_e - \mathbf{P}_p + t\mathbf{V}_e \\ \mathbf{V}_p &= \frac{\mathbf{P}_e - \mathbf{P}_p + t\mathbf{V}_e}{|\mathbf{P}_e - \mathbf{P}_p + t\mathbf{V}_e|} \end{aligned} \quad (5.14)$$

The last step is to change to the coordinate system of the projector as in the previous section.

$$\begin{aligned}\theta_1 &= \tan^{-1} \frac{Y(\mathbf{V}_p)}{X(\mathbf{V}_p)} \\ \theta_2 &= \cos^{-1} \left( \mathbf{V}_p \cdot -\hat{\mathbf{Z}} \right)\end{aligned}\tag{5.15}$$

Using Equation 5.8, the image coordinates can be found.

## Chapter 6

# Video Streaming

Recall that our goal is to transmit multiple live video streams across the network. A number of issues has to be considered when choosing a streaming model which satisfy the quality of service (QoS) requirements. QoS may be described in a variety ways, but we can characterize three different service requirements for tele-immersive data:

- *bandwidth* - it represents a raw throughput requirement, and it is typically specified in terms of constant and peak bit rates. Video data has well defined bandwidth requirements and recent network protocols such as IPv6 and ATM provide support for constant bit rate services.
- *reliability* - reliability represents the extent to which the data type is robust in bearing packet loss. Given a fundamentally unreliable delivery protocol such as IP, reliable delivery always add extra cost.
- *latency and jitter* - latency represents the delay associated with transmitting data from one site to another while jitter represents the variation in latency. These are significant problems over unreliable network (such as IPv4), which drop packets during period of high traffic rate.

## 6.1 Video Compression

As mentioned in Chapter 2, the common video compression standards are the H.261/H.263, MPEG1, MPEG2, MPEG4 and Motion JPEG. In our system, it is not our goal to develop a new video compression technique but to adopt off-the-shelf components to accomplish the work. Depending on different network condition, different video compression techniques may suit well. For example, for pre-recorded video, it is better to use MPEG encoding to achieve the highest compression ratio. While in real-time applications, the fast compression and decompression requirements favor the use of Motion JPEG. For tele-conferencing applications which only a talking head appears in the video, using H.261/H.263 may be a good choice.

In order to favor different application requirements, our framework is independent of the video coding format. Rather than incorporating a particular video compression and decompression scheme into the framework, the streaming module provides a container with which different video codec can be inserted in.

## 6.2 Transport Protocol

There are quite a number of well defined protocols available in our current networking technologies, namely TCP, UDP, RTP, RTSP, etc. Similar to the video compression techniques, these protocols are suitable in different situations. For example, TCP may be suitable for video streaming across an unreliable network. UDP is suitable for local LAN communications where the loss and latency are relatively minimal. RTP and RTSP may be employed for applications which require more multimedia control (e.g. priority management).

Similar to the independence of video codec, our system also provides an interface to the transport protocol. Different transport protocols can be employed in our framework to suit different application requirement.

## 6.3 Latency and Jitter Control

In video stream, latency expresses how much time it takes for data to get from one designated point to another. The sources for such latency include:

- *Propagation* - packets when transmitted over the physical medium requires time.
- *Intermediate nodes* - intermediate nodes such as router or gateway takes time to examine and possibly change the header of a packet
- *Other processing time* - encoding and decoding of data may introduce extra delay.

For live video streaming, the latency should be kept as short as possible. However, as the video is displayed in remote site and the viewer does not have contact with the real remote environment, viewer will not know about the time difference unless the application is "remote control".

When transmitting video over the network, besides the problem of latency, it suffers from the non-constant delay and packet loss. This results in a jerky appearance which affects the visual quality[47]. In the absence of jitter and packet loss, video frames can be played once received, resulting in a smooth playback as depicted in Figure 6.1a. However, in the presence of jitter, arrival time of frames vary as shown in Figure 6.1b. User would see the frozen image of the frame 2 until frame 3 arrived. Frame 3 would be displayed only in a very short in order to preserve the timing for the subsequent

frame 4.

In the presence of packet loss, some frames will not even arrive at the receiver as shown in Figure 6.1c. Frame 3 losses during transfer. receiver. Viewer would see a frozen image as frame 2 will be displayed for a much longer time. When frame 4 arrives, it will be displayed and viewer would see a jerky transition. As found by Mark[47], even low jitter will severely affect the perceptual quality<sup>1</sup>. Therefore, it is necessary to avoid jittering and to maintain a short latency at the same time.

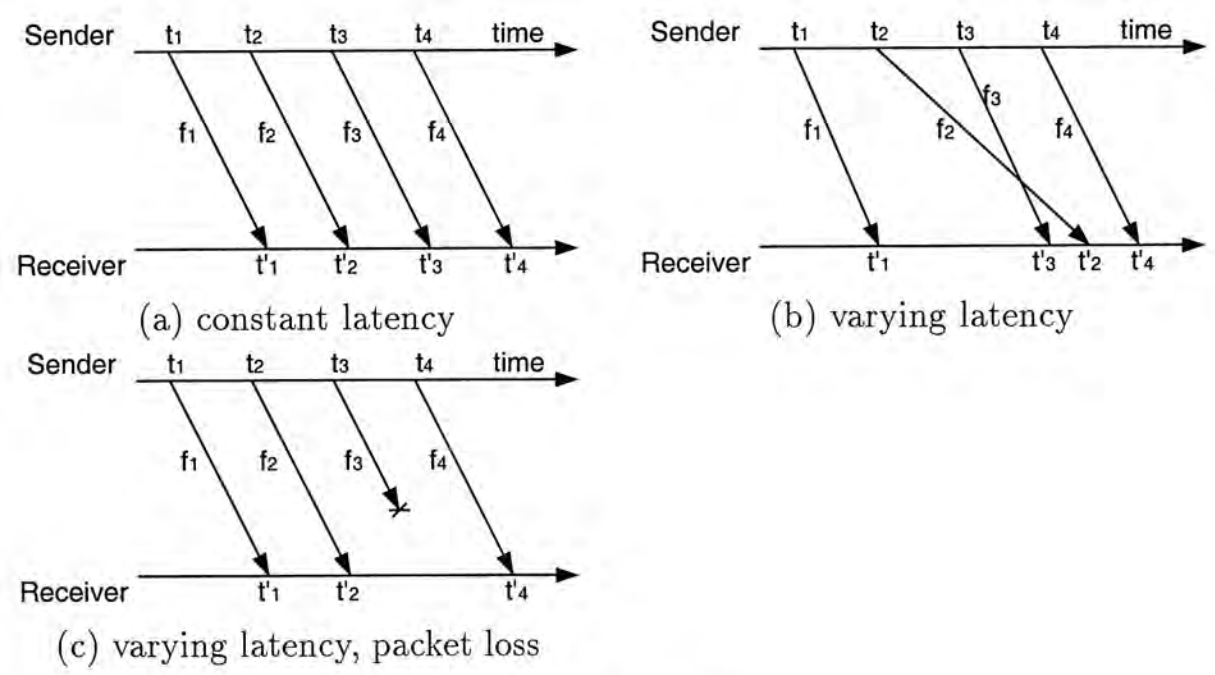


Figure 6.1: Jittering

In order to reduce the effect of jitter and latency when streaming the video, a simple jitter control scheme[48] is employed. In that scheme, the source machine gives timestamp to each of the media frames. Upon receiving the data in the target machine, knowing the timestamp and the maximum delay, buffers them until the time it has to be transferred to the stitching modules and the display module.

<sup>1</sup>The term *perceptual quality* means a quantitative measure of video quality from the user perspective.

This control scheme has made three assumptions[48, 49]:

1. The maximum delay of the network is bounded.
2. There are enough buffer space on receiver side.
3. The clocks of sender side and receiver side are synchronized.

These assumptions can be satisfied by current technologies. In doing network transmission, each packet can be tagged the time-to-live so that delay of a packet can be bounded. Current memory size is in the order of hundreds of megabytes which is more than enough for the buffer space usage. The assumption of a universal time source for clock synchronization is reasonable. The Network Time Protocol (NTP)[50] has been widely tested on the Internet and is capable of distributing time accurately to the millisecond. It is currently trying the possibility of distributing time as accurate as microsecond. Since video data has a period in the order of tens of millisecond. NTP is enough for having a synchronized clock for our distributed system.

This scheme compensates for jitter at the expense of latency. Transmitted frames are buffered in memory by the receiver, allowing each frame to be displayed with a constant latency, in order to achieve a steadier stream. However, the added latency from buffering can be disturbing for interactive applications as in our system. So, a buffer size is chosen to give the best visual quality while keep the latency minimal.

To deal with packet loss in the case of using UDP or RTP as the transport protocol, a simple approach is used. Rather than doing a re-transmission, the loss packet is ignored. Previous frame is used as a replacement for the mosaic composition. As the time difference between consecutive frames is small, little difference between the images is expected. The amount of visual artifact will not be significant. If loss packets are retransmitted, the amount

of delay will be very great. For interactive applications, obsoleted frames are no longer useful. Therefore, using previous frame as a replacement is better than retransmitting the loss frames.

## 6.4 Synchronization

Since our video capture units are distributed in different areas and they capture video asynchronously. It is necessary to synchronize the captured video streams as to provide video frames of the same instance to the stitching module. As mentioned in Chapter 4, without synchronization between video streams, the generated mosaic video will be erroneous.

There are various synchronization mechanism[51, 52, 53] for distributed systems. Rather than using sophisticated approach in achieving the synchronization, a simple approach is used. As mentioned before, the video frames from each streams are timestamped. Rather than using the logical timestamp (as in RTP), the real world time is used[54, 55]. At the receiver side, frames from the multiple streams are then ordered according to the timestamp. This simple approach assumes that the clocks of all the remote capture units are synchronized. This can be achieved by NTP as mentioned before.

## Chapter 7

# Implementation and Results

We have implemented several computer programs to verify the feasibility of our framework. This chapter will show some results of these experiments.

### 7.1 Video Capture

We start with the video capture module which provides the input data. In our current setup, eight CCD cameras<sup>1</sup> which provide PAL signals are fed into two independent quad mergers<sup>2</sup>. The outputs of the quad mergers are digitized by video capture boards<sup>3</sup> using BrookTree Bt878 chipset. Figure 7.1 shows one of our camera group which consists of four cameras, a quad merger and a computer.

Our video capture program (Figure 7.2) is developed on ordinary PCs with Windows 98. There are currently two main video capture software architectures supported. DirectShow is a relatively new one which is founded on a series of layers which allow hardware abstraction. With this architecture, components called filters are connected together to form a filter graph. Data is transformed as it is passing through the filters. When the data goes

---

<sup>1</sup>S-688C Security CCD Camera, Super Electronics Limited

<sup>2</sup>VP-6400 Real-Time DSP Color Quad, Taiwan Denpo Company Limited

<sup>3</sup>MachTV, MachSpeed Limited and Video Wonder Pro II, KYE Systems Corporation

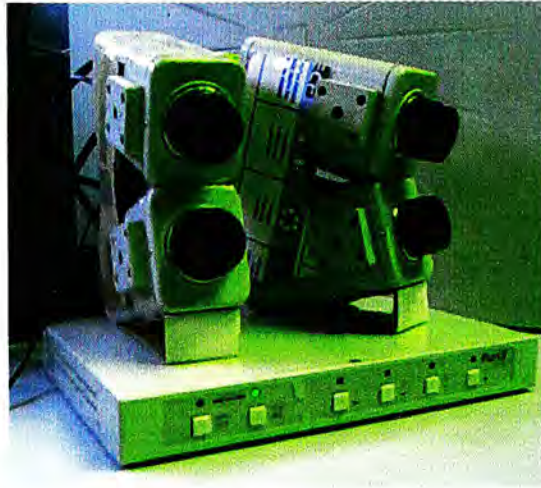


Figure 7.1: Video capture module

to the final rendering filter, the transformed data is displayed. Using DirectShow, it allows easy integration of other technologies like DirectX or other using the Microsoft COM model.

The other video capture application development architecture is the Video for Window (VfW) which is a set of well-established technologies. VfW was first release in 1992 and was optimized for capturing movies to disk. Though it is to be replaced by DirectShow, we still use it as our video capture for the following reasons:

1. The programming interface of VfW is stable while DirectShow is still preliminary and is under revision.
2. It allows the video capture module to be developed separately while DirectShow requires the complete filter graph be developed.
3. VfW architecture is more intuitive which other operating systems provide a similar interface.

Using our video capture program, it can attain a capture rate of as high as 120 frame per seconds. A video capture module for Linux has also been developed using the Video-4-Linux and BTTV interface.



Figure 7.2: User interface of the video capture program

## 7.2 Video Streaming

In the implementation of the streaming module, it is developed as a container of different video codecs. Also, it allows different system parameters to be changed to suit a particular network configuration.

### 7.2.1 Video Encoding

The streaming module is developed as a container of video codecs. Currently, our streaming module supports a variety of video codecs. In the first testing phase, we implemented a MPEG1 video codec using the Dalí multimedia software library[56] and a Motion-JPEG video codec using the Intel JPEG Library. These two video codecs are developed without extended optimization and are expected not to work very well.

Table 7.1 shows the performance of using different video codecs. The video codecs are tested by compressing a video stream of frame size 512 pixels times 512 pixels under a machine of Pentium II 400MHz. It is obvious that all codecs except Motion JPEG can compress at an interactive frame

Codec		Avg. FPS	Compression Ratio
MPEG	Our implementation <sup>a</sup>	7.143	62.85
	DivX4 ;-) <sup>b</sup>	0.915	54.89
	MPEG-4 V2/V3 <sup>c</sup>	8.479/8.572	113.10/112.93
MJPEG	Our implementation <sup>d</sup>	11.473	16.72
	Lead MJpeg <sup>e</sup>	17.922	19.43
	PICVideo MJpeg <sup>f</sup>	24.256	16.63
Misc.	Cinepak	0.680	23.25
	Indeo®R5.06	1.250	34.04
	Indeo®Interactive	0.866	45.40

Table 7.1: Comparison of video codec

<sup>a</sup>Dali VM, <http://www.cs.cornell.edu/dali>  
<sup>b</sup>OpenDivX Project, <http://www.projectmayo.com>  
<sup>c</sup>Microsoft Corporation MPEG-4 Video Codec  
<sup>d</sup>Intel JPEG Library Version 1.5, <http://developer.intel.com/software/products/perflib/ijl>  
<sup>e</sup>Lead Technologies Incorporation, <http://www.leadtools.com>  
<sup>f</sup>Pegasus Imaging Corporation, <http://www.jpg.com>

rate of around 12 fps. Users who watch the video at such frame rate, will not feel the discontinuity of the motions and the interaction of the scene. This performance is normally acceptable in most applications. For Motion-JPEG, it can achieve a real-time compression and decompression of around 24 fps. This real-time performance provides viewers additional fidelity of the virtual scene.

Although Motion-JPEG can achieve a high frame rate, it is only suitable for high network bandwidth environment. The compression ratios do not get as high as other compression schemes. So, in network with low bandwidth, other codecs are preferred.

MPEG and H.261/H.263 all use inter-frame compression method which reduce the redundant informations between consecutive frames. As extra redundancy is removed, compressed data size will be smaller, and hence higher compression ratio. This inter-frame compression, however, requires

extra computation to find the redundant information. With higher computation requirement, the rates of encoding and decoding are lower as expected. Motion-JPEG uses an elementary approach that employs the still frame compression technique on a frame-by-frame basis. The compression that can be achieved by such approach is limited. No temporal redundancies are removed that save the precious computation power for the fast compression. As a result, higher rate of compression can be obtained. In our comparison, all the computation is done in software which may vary greatly by the implementation. The PICVideo MJPEG is heavily optimized by using the MMX technology and by assembly programming; Dali's MPEG1 is not optimized at all; others are more or less optimized.

Instead of using software for the encoding, hardware codec may be employed to enhance the performance. Many MPEG compression cards can achieve more than 30 frame per seconds with very high quality. It is desirable to replace the software encoding with the hardware encoding technology.

### 7.2.2 Streaming Protocol

Our system is running on a 100 MBits Ethernet network. All the host machines are directly connected to the network with no intermediate nodes in between. Table 7.2 shows the properties of the network. The figures are obtained by sending 10,000 ICMP messages. The transmission delay is assumed to be half of round trip time (RTT). Thus, it has a maximum value of 48 ms.

$RTT_{max}$	96ms
$RTT_{min}$	9ms
$RTT_{avg}$	10ms
$Loss_{avg}$	0.233%

Table 7.2: Network properties

This network is quite reliable with a low rate of packet loss. It is therefore not necessary to use TCP/IP as the transport protocol as this may introduce extra processing. Rather, using the UDP/IP protocol can handle this situation well. Using both TCP or UDP can give similar video streaming quality.

## 7.3 Implementation Results

Our system has been tested for both indoor scene and outdoor scene. Both the results are satisfactory.

### 7.3.1 Indoor Scene

Figure 7.3 shows the video capture module for capturing indoor scene. Since the objects in the indoor environment are close to the camera, the cameras should be placed as close together as possible to avoid the parallax.



Figure 7.3: camera arrangement for indoor scene

Figure 7.4 shows an initial image capture for the preprocessing stage. Point pairs are manually entered to the image correspondence.

Using the steps as mentioned in Chapter 4, the input images are stitched together to form a large field of view image as shown in Figure 7.5.

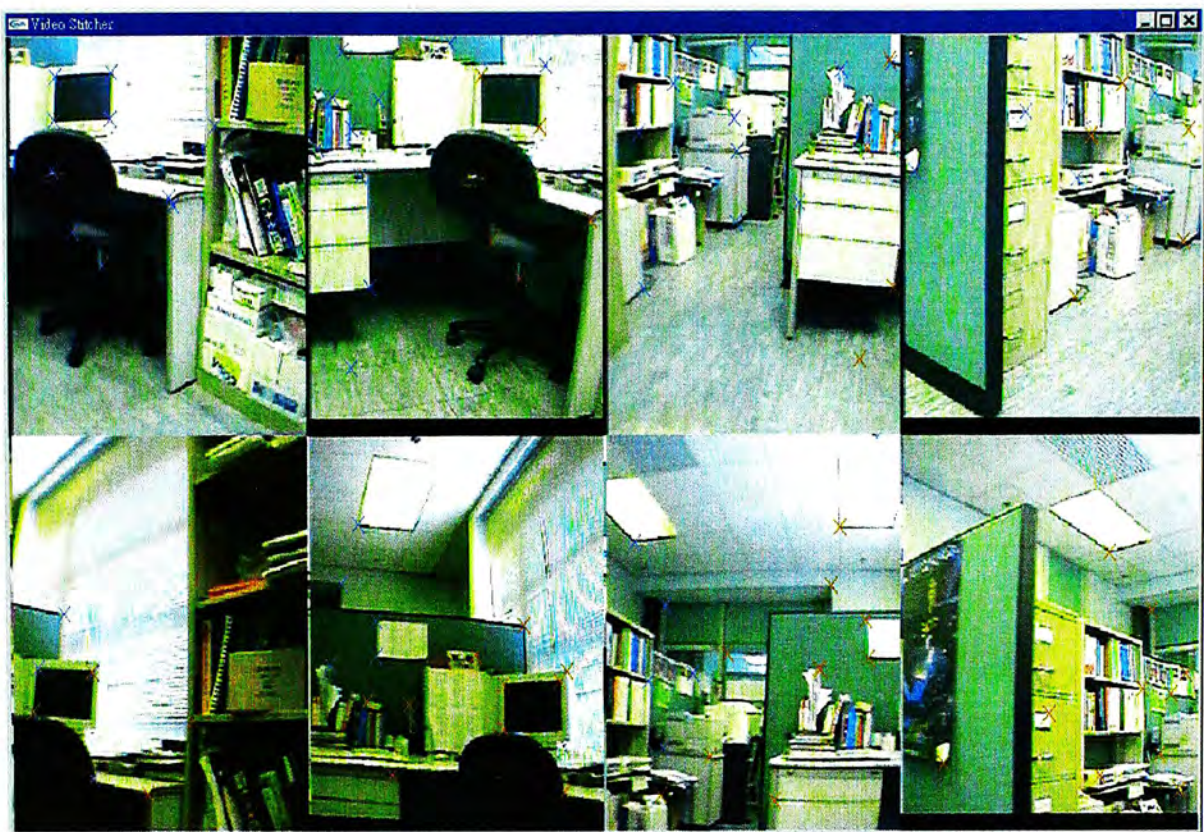


Figure 7.4: Correspondence is identified



Figure 7.5: Input Images are stitched together

This image is then warped according to the projector characteristics. This is shown in Figure 7.6.



Figure 7.6: Stitched image is warped

This warped image is projected onto VisionStation as depicted in Figure 7.7.

### 7.3.2 Outdoor Scene

In outdoor environment, objects are expected to be far away. The cameras can be separated for larger distance without significant error due to parallax. The cameras are arranged as depicted in Figure 3.4 in Chapter 3. The two camera groups are separated by approximately 3 meters. With this large displacement between the cameras, the video streams can still be combined with little visual artifact.

## 7.4 Evaluation

As shown in the figures in previous section, the generated panoramic images are not sharp enough. In some case, ghosting effect can be seen. The problem

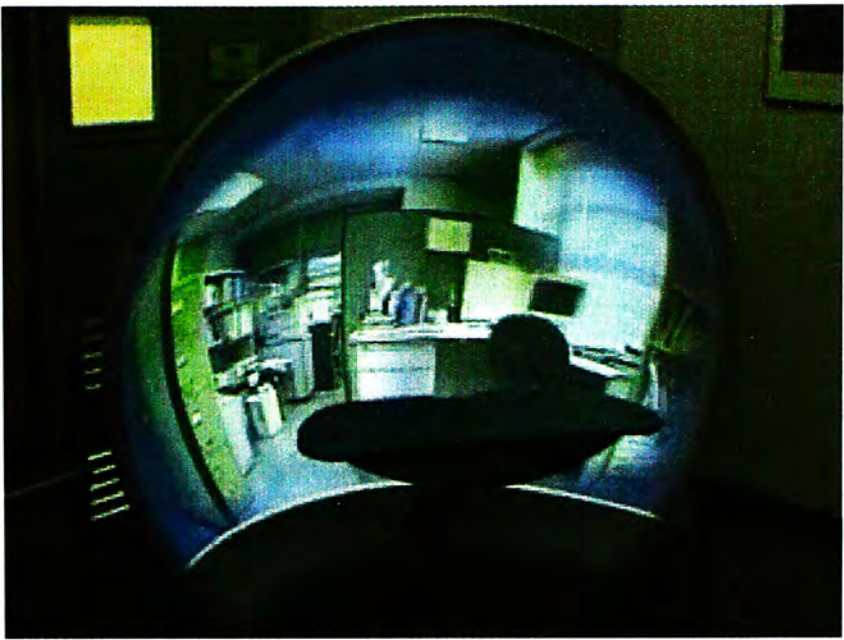


Figure 7.7: Warped image is projected onto VisionStation



Figure 7.8: Outdoor image sources



Figure 7.9: Outdoor image sources



Figure 7.10: Outdoor image sources

is mainly due to three reasons:

1. When capturing the live video streams through the video quad mergers, each single video stream has a resolution of  $256 \times 256$ . This low resolution prevents the composing of high resolution panoramic video streams.
2. In transmitting the video across the network, the process of compression and decompression greatly decreases the image quality.
3. There are always some error in the stitching process. The result cannot be perfectly apply on all the subsequent videos frames for rendering.

For the varying intensity, the indoor scene does not produce noticeable intensity inconsistency. This is because in an indoor environment, the lighting condition is relatively stable. Different directions have similar lighting intensity which allow the generation of seamless panoramic images. For outdoor scene, the lighting conditions vary greatly in different part of the scene. For a camera looking at a bright object such as the sun (Figure 7.11), the captured image will be very dark except the bright object. Blending this image with other will always result in noticeable intensity difference.



Figure 7.11: Capturing at the dawn

The captured images look fine at first, but eventually the capture becomes jittery after the system has been running for a while. This problem is not related to the streaming policy but because of the video capture module. As the system is running, heat build up on the video capture board which wear down the performance of the video capturing process.

## Chapter 8

# Conclusion

This chapter will give a summary of this project and conclude the thesis with possible improvement and future research aspect.

### 8.1 Summary

The goal of this work is to develop a tele-immersion system which allows viewers who situated in the office/control room to freely see the surroundings as if he/she is located in remote area. To develop such a system, we have divided it into a number of modules. They include the video capture module, video streaming module and the video mosaic module. Each module has quite a different set of challenges.

In summary, we have achieved our goal of developing a framework for tele-immersion systems. Eight live video streams are captured by the CCD camera clusters. The captured streams are streamed across the network to the destination computer where they are combined to become a single wide field of view video stream. This live generated vido stream is projected onto an immersive display, the VisionStation, which provides users the first person tele-immersive experience. Our implementation demonstrates the feasibility and practicability of this system framework in applying real world problems.

## 8.2 Future Directions

The current implementation of the system have some rooms for further improvement.

Currently, our system only uses eight cameras which cover only about half of the viewing sphere. If more cameras can be included in the system, more viewing directions will be allowed. The viewer can freely pan around which can improve the immersiveness and also the usability of the system.

We have implemented the whole system for testing purpose only. The ultimate goal is to build a ready-to-use system, from acquiring video to producing the panoramic video. In addition, we can improve our implementation in various aspects like a more user-friendly interface.

Another deficiency of the system is on color management. Each camera has its own automatic gain control which adjust the brightness of the pictures. Cameras facing different directions may have quite different gain value. Video frames of different cameras, when combined together, may have very noticeable seam between. Our system tries to minimize this artifact using a weighed blending function. This, however, can only reduce the artifact by a little bit. Possible solutions, such as recovering the high dynamic range radiance maps or using programmable exposure cameras, may require further investigation.

We have an assumption that all the objects in the scene is far away and the optical center of the cameras coincide. These make our stitching algorithm simpler. These assumptions, however, may not be valid in all the situations. In that case, more involved models and algorithms, such as real-time recovery of three dimensional geometry from pictures may be necessary. However, that problem is still open and requires more intensive research.

We are using the VisionStation for displaying. In future, we may extend

the displaying device to a normal room using a casual placement of projectors and projector screens as introduced by Raskar[30, 45].

For a tele-immersive application, only visual information may not be enough. It is possible to include a surround sound system which will greatly enhance the applications. We hope that real applications will be in used in the near future.

# Appendix A

## Parallax

Parallax is the difference in apparent direction of an object as seen from two different points not on a straight line with the object. The first assumption of our system is that the optical centers of all cameras are at the same point. If a 3D point has parallax less than half of the minimum angle subtended by a pixel, then when that point is projected onto the cameras image planes, the pixel coordinates will deviate less than half a pixel as if the optical center of all the cameras are at the same point. Using this constraint, the minimum distance of a point from the camera clusters can be found so as to satisfy the assumption.

Figure A.1 shows the case of two ideal cameras. Let  $A_e$  be the minimum angle subtended between two pixels and  $D_e$  be the distance between the optical centers of the cameras  $C_1$  and  $C_2$ .

The following inequality can be established:

$$A < \frac{A_e}{2} \tag{A.1}$$

Since

$$\tan \frac{A}{2} \approx \frac{D_e/2}{D}$$

the inequality A.1 can be rewritten as

$$\frac{D_e/2}{D} < \tan \frac{A_e}{4}$$

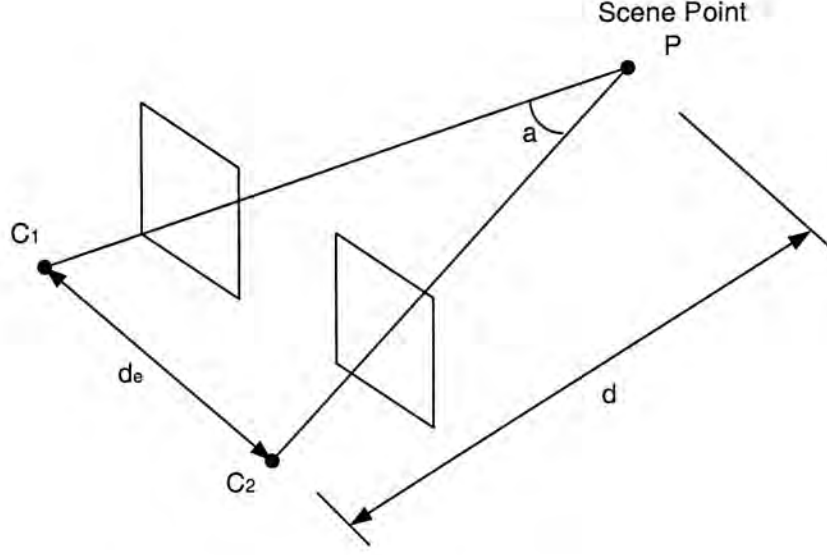


Figure A.1: Parallax of two cameras

As  $A_e$  is usually very small,  $\tan \frac{A_e}{4}$  is approximately equal to  $\frac{A_e}{4}$ . Therefore,

$$\frac{D_e/2}{D} < \frac{A_e}{4}$$

$$D > \frac{2D_e}{A_e} \quad (\text{A.2})$$

$$(\text{A.3})$$

For the case of  $n$  cameras, the conditions are similar. Instead,  $D_e$  is replaced by the maximum distance between the optical center of any of the two cameras.

To find the minimum distance  $D_{min} = \frac{2D_e}{A_e}$ , it is necessary to obtain the values of the two unknowns.  $D_e$  can be obtained by direct measurement. For  $A_e$ , as focal lengths of the camera are found in the calibration step, the field of view is obvious and so is the value of  $A_e$ .

In our current setup,  $D_e$  is approximately 3 meters and  $A_e$  is about 0.00356 radian. Thus, the minimum distance  $D_{min}$  is about 1.684 kilometers. As the cameras are pointing at the outdoor scene, the distance of the objects are far more than  $d_{min}$ . Therefore, the assumption is valid in our case.

# Bibliography

- [1] S. E. Chen, “QuickTime VR — an image-based approach to virtual environment navigation,” *Proceedings of SIGGRAPH 95 Conference*, 1995.
- [2] *IPIX*. <http://www.ipix.com/>.
- [3] R. Swaminathan and S. K. Nayar, “Polycameras: Camera clusters for wide angle imaging,” Tech. Rep. CUCS-012-99, Columbia University Technical Report, New York, 1999.
- [4] S. K. Nayar, “Omnidirectional video camera,” in *Proceedings of the 1997 DARPA Image Understanding Workshop*, May 1997.
- [5] J. Gluckman, S. K. Nayar, and K. J. Thoresz, “Real-time omnidirectional and panoramic stereo,” in *Proceedings of the 1998 DARPA Image Understanding Workshop*, (California, USA), November 1998.
- [6] C. Fraleigh, B. Johanson, H. Kesavan, B. Nelson, M. Schroepfer, L. Y. Wei, and B. Wilburn, “Video panoramas project.” <http://graphics.stanford.edu/courses/cs448-98-fall/dtv/panoramas.html>, 1998.
- [7] J. Lengyel, “The convergence of graphics and vision,” *Computer, IEEE Computer Society Magazine*, pp. 46–53, July 1998.

- [8] R. Szeliski, "Video mosaics for virtual environments," *Virtual Reality*, pp. 22–30, March 1996.
- [9] R. Szeliski and H.-Y. Shum, "Creating full view panoramic image mosaics and texture-mapped models," in *Proceedings of SIGGRAPH 97*, pp. 251–258, August 1997.
- [10] L. G. Brown, "A survey of image registration techniques," in *ACM Computing Surveys*, vol. 24, pp. 325–376, 1992.
- [11] C. D. Kuglin and D. C. Hines, "The phase correlation image alignment method," in *Conference on Cybernetics and Society*, pp. 163–165, September 1975.
- [12] B. S. Reddy and B. N. Chatterji, "An FFT-based technique for translation, rotation, and scale-invariant image registration," in *IEEE Transactions on Image Processing*, vol. 5, 1996.
- [13] J. Davis, "Mosaics of scenes with moving objects," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 1998.
- [14] P. Dani and S. Chaudhuri, "Automated assembling of images: Image montage preparation," in *Pattern Recognition*, vol. 28, pp. 431–445, March 1995.
- [15] D. L. Milgram, "Adaptive techniques for photo mosaicing," in *IEEE Transactions in Computers*, vol. C, pp. 1175–1180, 1977.
- [16] R. Szeliski, "Image mosaicing for tele-reality applications," Tech. Rep. CRL 94/2, Digital Equipment Corporation Cambridge Research Lab, May 1994.

- [17] M. G. Gnzaalez, P. Holifield, and M. Varle, "Improved video mosaic construction by accumulated alignment error distribution," in *Proceedings of the British Machine Vision Conference*, 1998.
- [18] H. Y. Shum and R. Szeliski, "Panoramic image mosaics," Tech. Rep. MST-TR-97-23, Microsoft Research, 1997.
- [19] H. S. Sawhney and R. Kumar, "True multi-image alignment and its application to mosaicing and lens distortion correction," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 450–456, 1997.
- [20] S. Peleg and J. Herman, "Panoramic mosaics by manifold projection," in *Proceedings of Computer Vision and Pattern Recognition*, pp. 338–343, June 1997.
- [21] H. S. Sawhney, S. Hsu, and R. Kumar, "Robust video mosaicing through topology inference and local to global alignment," in *Proceedings of the European Conference on Computer Vision*, 1998.
- [22] S. Gümüştekin and R. W. Hall, "Mosaic image generation on a flattened gaussian sphere," in *Proceedings of IEEE Workshop on Applications of Computer Vision*, pp. 50–55, 1996.
- [23] P. J. Burt and E. H. Adelson, "A multiresolution spline with application to image mosaics," *ACM Transactions on Graphics*, vol. 2, no. 4, pp. 217–236, 1983.
- [24] D. Browning, C. Neira, C. Sandin, and T. DeFanti, "The cave automatic virtual environment: Projection-based virtual environments and disability," in *Proceedings of the First Annual International Conference, Virtual Reality and People with Disabilities*, January 1993.

- [25] C. Cruz-Neira, D. J. Sandin, and T. A. DeFanti, "Surround-screen projection-based virtual reality: The design and implementation of the cave," *Proceedings of SIGGRAPH 93*, pp. 135–142, August 1993.
- [26] Alternate Realities Corporation. <http://www.elumens.com/>.
- [27] Fakespace Systems. <http://www.fakespacesystems.com/>.
- [28] Panoram Technologies, Inc. <http://www.panoramtech.com/>.
- [29] Trimension Systems Ltd. <http://www.trimension-inc.com/>.
- [30] R. Raskar, M. S. Brown, R. Yang, W. C. Chen, G. Welch, H. Towles, B. Seales, and H. Fuchs, "Multi-projector displays using camera-based registration," in *Proceedings of IEEE Visualization 1999*, (San Fransisco, CA), pp. 161–168, October 24-29 1999.
- [31] P. J. Shenoy, P. Goyal, and H. M. Vin, "Issues in multimedia server design," *Computing Surveys*, vol. 27, no. 4, pp. 636–639, 1995.
- [32] J. Hunter, V. Witana, and M. Antoniadis, "A review of video streaming over the internet," Tech. Rep. TR97-10, Distributed System Technology Centre, August 1997.
- [33] D. Anderson, S. Tzou, R. Wahbe, R. Govindan, and M. Andrews, "Support for live digital audio and video," in *Proceedings of 10th International Conference on Distributed Computing Systems*, (Paris, France), pp. 54–61, May 1990.
- [34] Request for Comments 1889, January 1996.
- [35] "Real time streaming protocol (rtsp)." Request for Comments 2326, April 1998.

- [36] A. Majumder, G. Meenakshisundaram, W. B. Seales, and H. Fuchs, "Immersive teleconferencing: A new algorithm to generate seamless panoramic imagery," in *ACM Multimedia*, (Orlando, FL), pp. 169–178, 30 October – 5 November 1999.
- [37] P. F. Sturm and S. J. Maybank, "On plane-based camera calibration: A general algorithm," in *Proceedings of Computer Vision and Pattern Recognition*, pp. 432–437, 1999.
- [38] Z. Zhang, "Flexible camera calibration by viewing a plane from unknown orientations," in *International Conference on Computer Vision*, (Corfu, Greece), pp. 666–673, September 1999.
- [39] D. E. Stevenson and M. M. Fleck, "Robot aerobics: Four easy steps to a more flexible calibration," in *International Conference on Computer Vision*, (Cambridge MA), pp. 34–39, 1995.
- [40] J. Heikkila and O. Silven, "A four-step camera calibration procedure with implicit image correction," in *Proceedings of the Conference on Computer Vision and Pattern Recognition*, 1997.
- [41] M. L. Heliig and E. C. del Futuro, "The cinema of the future," *Presence*, vol. 1, no. 3, pp. 279–294, 1992.
- [42] D. R. Schikore, R. A. Fischer, R. Frank, R. Gaunt, J. Hobson, and B. Whitlock, "High-resolution multiprojector display wall," *IEEE Computer Graphics and Applications*, vol. 20, pp. 38–44, July/August 2000.
- [43] R. Raskar, "Immersive planar display using roughly aligned projectors," in *IEEE Virtual Reality 2000*, March 18–22 2000.

- [44] M. Hirose, T. Ogi, and T. Yamada, "Integrating live video for immersive environments," *IEEE MultiMedia*, vol. 6, pp. 14–22, July–September 1999.
- [45] R. Raskar, G. Welch, M. Cutts, A. Lake, L. Stesin, and H. Fuchs, "The office of the future: A unified approach to image-based modeling and spatially immersive displays," *Proceedings of SIGGRAPH 98*, pp. 179–188, July 1998.
- [46] Alternate Realities Corporation, "Tilttable hemispherical optical projection systems and methods having constant angular separation of projected pixels." United States Patent, 9 June 1998. Patent Number: 5762413.
- [47] M. Claypool and J. Tanner, "The effects of jitter on the perceptual quality of video," in *ACM Multimedia*, (Orlando, FL), October 30–November 5 1999.
- [48] C. Partridge, "Isochronous applications do not require jitter-controlled networks." Request for Comments 1257, September 1991.
- [49] D. Ferrari, "Design and applications of a delay jitter control scheme for packet-switching internetworks," in *Proceedings of the second International Conference on Network and Operating System Support for Digital Audio and Video*, 1991.
- [50] D. Mills, "Measured performance of the network time protocol in the internet system." Request for Comments 1128.

- [51] S. Ramanathan and P. V. Rangan, "Continuous media synchronization in distributed multimedia systems," in *Proceedings of the 3rd International Workshop on Network and Operating System Support for Digital Audio and Video*, (La Jolla, CA, USA), pp. 328–335, November 1992.
- [52] S. Tasaka and Y. Ishibashi, "Single-stream versus multi-stream for live media synchronization," in *IEICE Transactions on Communications*, vol. E81 of 11, 1988.
- [53] D. Jaiswal, "Media coding and distribution over unreliable networks : Some issues," *Journal of The Institution of Electronics and Telecommunication Engineers*, pp. 399–412, September-October 1999.
- [54] L. Lamport, "Time, clocks, and the ordering of events in a distributed system," *Communications of the ACM*, vol. 21, no. 7, pp. 558–565, 1978.
- [55] T. Wahl and K. Rothermel, "Representing time in multimedia systems," in *International Conference on Multimedia Computing and Systems*, pp. 538–543, 1994.
- [56] W. T. Ooi, B. Smith, S. Mukhopadhyay, H. H. Chan, S. Weiss, and M. Chiu, "The dali multimedia software library," in *1999 SPIE Multimedia Computing and Networking*, (San Jose, CA), January 25–27 1999.
- [57] K. S. Lee, Y. F. Fung, K. H. Wong, S. H. Or, and T. K. Lao, "Panoramic video representation using mosaic image," in *CISST'99*, (Las Vegas, USA), pp. 390–396, June 1999.
- [58] P. E. Debevec and J. Malik, "Recovering high dynamic range radiance maps from photographs," in *Proceedings of SIGGRAPH 97* (T. Whitted, ed.), (Los Angeles, California), pp. 369–378, August 1997.

- [59] V. N. Peri and S. K. Nayar, "Generation of perspective and panoramic video from omnidirectional video," in *Proceedings of DARPA Image Understanding Workshop*, pp. 243–245, 1997.
- [60] J. Mulligan and K. Daniilidis, "View-independent scene acquisition for tele-presence," in *Proceedings of International Symposium on Augmented Reality*, October 2000.
- [61] N. Greene and P. S. Heckbert, "Creating raster omnimax images from multiple perspective views using the elliptical weighted average filter," *IEEE Computer Graphics and Applications*, pp. 21–27, June 1986.
- [62] D. L. Stone and K. Jeffay, "An empirical study of delay jitter management policies," *Multimedia Systems*, vol. 2, pp. 267–279, January 1995.
- [63] Y. Xiong and K. Turkowski, "Creating image-based VR using a self-calibrating fisheye lens," in *Conference on Computer Vision and Pattern Recognition*, 1997.
- [64] M. Jethwa, A. Zisserman, and A. Fitzgibbon, "Real-time panoramic mosaics and augmented reality," in *Proceedings of the 9th British Machine Vision Conference*, (Southampton, UK), pp. 852–862, BMVA Press, September 1998.



CUHK Libraries



003871881