uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Yue Xing

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.A.Sc. (Electrical Engineering)

GRADE / DEGREE

School of Information Technology and Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Towards a 3D Immersive Teleconferencing System – Design and Implementation

TITRE DE LA THÈSE / TITLE OF THESIS

Prof. Georganas

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

Prof. Dubois

Prof. Joslin

Gary W. Slater

Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

TOWARDS A 3D IMMERSIVE TELECONFERENCING SYSTEM - DESIGN AND IMPLEMENTATION

by

Yue Xing

A thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the M.A.Sc. degree in Electrical Engineering

Ottawa-Carleton Institute for Electrical & Computer Engineering
School of Information Technology and Engineering
Faculty of Engineering
University of Ottawa

Your file Votre référence
ISBN: 978-0-494-49298-7
Our file Notre référence
ISBN: 978-0-494-49298-7

Abstract

Videoconferencing is becoming attractive for geographically distributed team collaboration. A 3D videoconference provides immersive telepresence and natural representation of all participants in a shared virtual meeting space to enhance quality of human-centered communication. In this thesis, an initial effort towards a 3D immersive teleconferencing system is presented, where the most advanced hardware and software technologies are applied, such as scene acquisition, modeling, rendering, tracking and stereo projective display. Especially the Exact Polyhedral Visual Hull algorithm has been used and proven to be robust for real-time 3D reconstruction. Customer–built polarization-multiplexed displays can provide a significant immersive impression. The Real-time Transport Protocol is employed for real-time communication. Performance analysis demonstrates that the implementation is a feasible framework towards the ultimate goal of 3D collaborative telepresence.

Acknowledgments

First and foremost I would like to express my deepest gratitude to my advisor Prof. Nicolas D. Georganas for assigning me this exciting research topic with many devices. Without his efforts the completion of the thesis would not be possible. I am thankful for his efforts to help me in spite of his busy schedule.

I have had the great fortune of working with some very smart people during the course of my thesis. I would like to thank all my colleagues in the DISCOVER lab especially Dr. Xiaojun Shen, Francois Malric and Jilin Zhou. Dr. Shen's vision helped steer the project. Francois gave me a lot support on devices. Jilin's practical suggestions helped the implementation.

Then, I would like to thank Zhenxia Zhang for his practical help. I would also like to thank Jean-Sébastien Franco for his EPVH algorithm. His advice helped greatly in the project.

Finally, I would like to thank my family for their invaluable support from the other side of the earth and my girl friend Baifen Li for always being there for me.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

Videoconferencing is a set of interactive telecommunication technologies which allow multiple dispersed participants to interact with each other via bidirectional video and audio transmissions simultaneously. The idea of video conferencing and video telephony has fascinated researchers for nearly 100 years. In 1927, a video phone conversation with live video transmission over telephone lines from Washington D.C. to an auditorium in New York City was demonstrated. The Bell Picturephone shown in Figure1 used a mechanical Nipkow Disk for sampling the images and had a display of size 2 by 2.5 inches. In the sixties, a couple of different video phones were developed (an example shown in Figure 2). Unfortunately, these technologies never became well accepted due to the costs and limited availability.

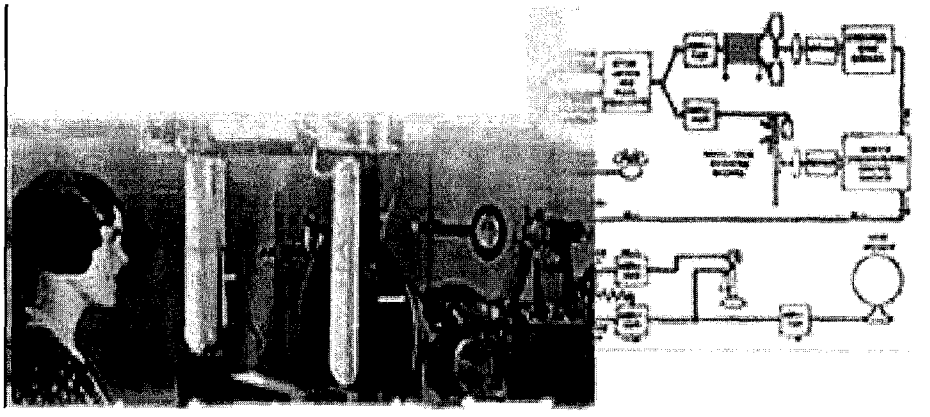


Figure 1: Bell's Picturephone system of 1927 [1]

Figure 2: Picturephone from 1964 [1]

With the introduction of digital computers, the increase of bandwidth capabilities and the development of presence engineering, video conferencing is rapidly advancing and regarded as a high-return investment for decision-making in a wide range of applications,

from international research projects, world-wide operating companies to global commerce, and many others. Related research can be found in the framework of the US Internet2 consortium, for instance the Access Grid (AG) [2], the virtual Rooms Video Conferencing Service (VRVS) [3], the Virtual Auditorium of Stanford University [4] or the Global Conference System [5]. Commercial examples are from Yahoo Messenger, NetMeeting to Plasma-Lift videoconference table [6] and Telepresence systems [7].

With MPEG-2 or H.263 2D video codecs, these systems offer high-quality audio and video equipment with presence capabilities for different applications like telelearning, teleconferencing and telecollaboration. Nevertheless they are limited in their support of natural human-centered communication. Body postures, subtle movements, eye contact, gaze direction, and room acoustics are often inefficient or misrepresented.

Human-centered communication in videoconferences means that every participant has his or her very individual perspective view of the conference scene – a feature that requires an interactive control of a virtual camera in a synthetic 3D world. This is a main issue of basic research on collaborative virtual environments (CVE). A common theme of all these efforts is to exploit the benefits of teleimmersion, often in such a way that the participants will have the impression of being presented in a shared virtual environment (as shown in Figure 3) suggesting spatial proximity, enabling a higher degree of natural interaction and effective collaboration. In other words, teleimmersion provides a realistic feeling that conferees share and inhabit the same space with remote colleagues.

Figure 3: Concept of an immersive shared virtual table video conferencing system [8]

In the last few years, much research work, which will be introduced in chapter 2, has been done towards this goal. In this thesis, the first steps towards the implementation of an immersive 3D video conferencing system are presented. This system is capable of live 3D scene reconstruction, view-dependent rendering and stereo display while operating between remote sites over gigabit Ethernet. The basic idea of the immersive 3D video conferencing is that 3D representations of the conferees are integrated into a shared virtual environment. Multi-views of Conferees captured by multiple cameras are sent to the remote site. 3D mesh models are reconstructed from 2D images. Virtual scenes are rendered and shown on 3D displays depending on the remote partner's viewpoint.

The 3D teleconferencing system currently operates in half-duplex mode, which means 3D acquisition and 3D display are not currently co-located. Audio communication which is an essential part of teleconferencing is not currently supported. As the system is only

tested over a LAN at DISCOVER lab, long distance communication performance needs to be investigated in the near future.

Immersive 3D video conferencing is a new interdisciplinary research field that links computer graphics, computer vision, image processing, and telecommunication. It also includes a lot of hardware knowledge about cameras and 3D displays.

1.2 Contributions

At the DISCOVER lab, there are two systems which are related to the 3D immersive teleconferencing. One is Zaxel's Virtual Viewpoint system [63] which live records the full 3D shape, texture, color and sound of moving real-world objects. It captures a subject using 12 surrounding cameras (each group of four cameras are driven by a slave computer) and converts the video images into 3D geometrical and textural information, then generates a new arbitrary viewpoint in 3D space based on an operator's input. Those three slave computers are connected to a master control workstation over Gigabit Ethernet. The control computer uses Zaxel provided software for the 3D-visualization, directing the slave computers to the different computational tasks needed for the generation of specific 3D views. As it can not reconstruct a 3D mesh model, the system can not show the captured 3D objects on a 3D display.

Another system is a circular polarization multiplexed display system named DIVINE (Desktop-Immersive Virtual and Interactive Networked Environment System). This

system was utilized to show computer made graphics, but never employed to show real time captured 3D objects. With 32 cameras, 7 displays and 7 computers, it provides two ideal ports for 3D Immersive Videoconferencing, but corresponding software is short. For technical reasons, I did not try to exclusively use this system for this thesis.

Given the goal and motivation of the 3D immersive teleconferencing, I designed a real-time teleimmersion system towards this goal based on the two foregoing systems. The main contributions can be summarized as follows:

Integrated, network distributed teleimmersion system: One server captures images from four cameras in the Zaxel room and processes them. Another server performs 3D reconstruction. One rendering cluster made up of three computers drives six projectors for stereo display in DIVINE. They are connected by Gigabit Ethernet. I combined state-of-the-art research components from computer vision, computer graphics and image processing to create a novel system designed to enhance the sense of immersion for teleconferencing tasks.

The Exact Polyhedral Visual Hull (EPVH) algorithm [38] is applied to reconstruct 3D models, which performs two times faster than the Image Based Visual Hull (IBVH) algorithm [37] used by Blue-C [19]. IBVH is pixel and depth based, while EPVH generates 3D mesh-based models. There are three advantages of using 3D mesh models. First of all, meshes can be rendered quickly by a graphics card. Secondly, 3D mesh models are easily combined with other 3D graphics. This offers a convenient and natural way for the conferees to collaborate with shared 3D objects. Thirdly, 3D mesh models

can be easily transferred over Internet using existing 3D graphics representation standards, such as VRML and X3D.

View-dependent texture mapping technique is employed, allowing the user to change the perspective of remote conferee according to the user's viewpoint. Viewpoint information is obtained by a magnetic sensor on the user's head.

Real-time Accurate Immersive Rendering and Display: By using three life-size stereoscopic displays to form a corner of a cube, we create a portal to a remote place with a compelling continuum between the local and remote sites. In contrast, most existing 3D conferencing systems are using 2D displays, such as VIRTUE [14], Coliseum [17] and TEEVE [21], or using only one stereo display, such as NTII [13], which provides participants no or limited sense of immersion. In stead of rendering still model on DIVINE, right after getting a new frame of the model with texture, the system renders and displays it in real time. Currently, the overall rendering speed is 3 frames per second.

Real-time 3D communication over gigabit Ethernet: Most applications, such as blue-C and low-cost Telepresence [22], treat 3D information as normal data and employ the Transmission Control Protocol (TCP) for the network transmission. In the presented system, 3D graphics are treated as streaming data and the Real-time Transport Protocol (RTP) is used, which make the system more suitable for real-time communication.

1.3 Thesis Outline

The rest of this thesis is organized as follows. As a background study, previous work and basic geometry are introduced in chapter 2. In chapter 3, the main technologies used in the 3D immersive video conferencing system are discussed. In chapter 4, the key aspects of the implementation are addressed. Chapter 5 describes the experimental results and analysis. Finally future work and conclusions are stated in chapter 6.

Chapter 2

Background

2.1 Previous Work

An early multiperson conference combining the round-table concept with at least semi-immersive display technology was the MAJIC system (multi-attendance joint interface for collaboration), proposed in 1994 by Keio University [9]. Life-size video of the remote users was shown on a semi-transparent curved screen. Eye contact and gaze perceiving were supported by careful placement of people, cameras and projectors.

Some years later, the University of Tokyo proposed the MONJUnoCHIE system [10]. In this system, a special semi-transparent display based on holographic optical elements was used to realize eye contact and gaze awareness. Both MAJIC and MONJUnoCHIE used separated windows or screens and 2D video to represent the remote partner. Thus, they did not scale well in terms of the number of participants.

Rather than assigning separated displays or windows to the different remote partners, the TELEPORT system moved further towards real immersive conferencing [11] and

implemented a shared virtual table. For the purpose of seamless integration of the videos into the virtual 3D environment, the partner's silhouette was segmented by means of chroma-or delta-keying. 3×2.25m full-wall projection allowed for stereoscopic viewing with passive glasses. A viewer tracking system determined the position of the local user to render images from the right viewpoint and to enhance the effect of a seamless transition between the real and virtual world.

Figure 4: Setup of NTII at UNC [13]

Although the TELEPORT used the concept of virtual 3D environments, the segmented videos that have been immerged into these environments were still limited to 2D representation. In contrast, the US National Tele-Immersion Initiative (NTII) proposed another approach that applied 3D video representation to a videoconferencing system based on so-called telecubicles [12] [13]. As shown in Figure 4, the remote participants appeared life size on separate stereo displays. The head position of the local user was tracked through sensors, and the stereo rendering was carried out with respect to the

tracker data; therefore, every participant can observe who is talking to whom or who is pointing at what. Voxel data based 3D video was derived from multiview images captured by the multi-baseline cameras set up and was sent to all remote sites. Dynamic voxel models could be viewed in stereo through shuttered glasses. However the NTII still used separated windows for the remote partners and thus did not scale well.

NTII used complex display and tracking devices. Users had to wear glasses to perceive 3D cues. To bypass this problem, VIRTUE (virtual team user environment, later called im.point immersive meeting point) went back to 2D video rendered via a virtual camera. Head tracker was removed by head motion parallax viewing algorithm. Scene depth information, 2D video and multimedia integration are coded using MPEG-4 [14] [15]. Similar solutions are the RealMeet room system and TeleSuite [16]. Other solutions, such as the Coliseum [17] and Gaze-2 [18], reduced the complexity of immersive videoconferencing to usual desktop working environments.

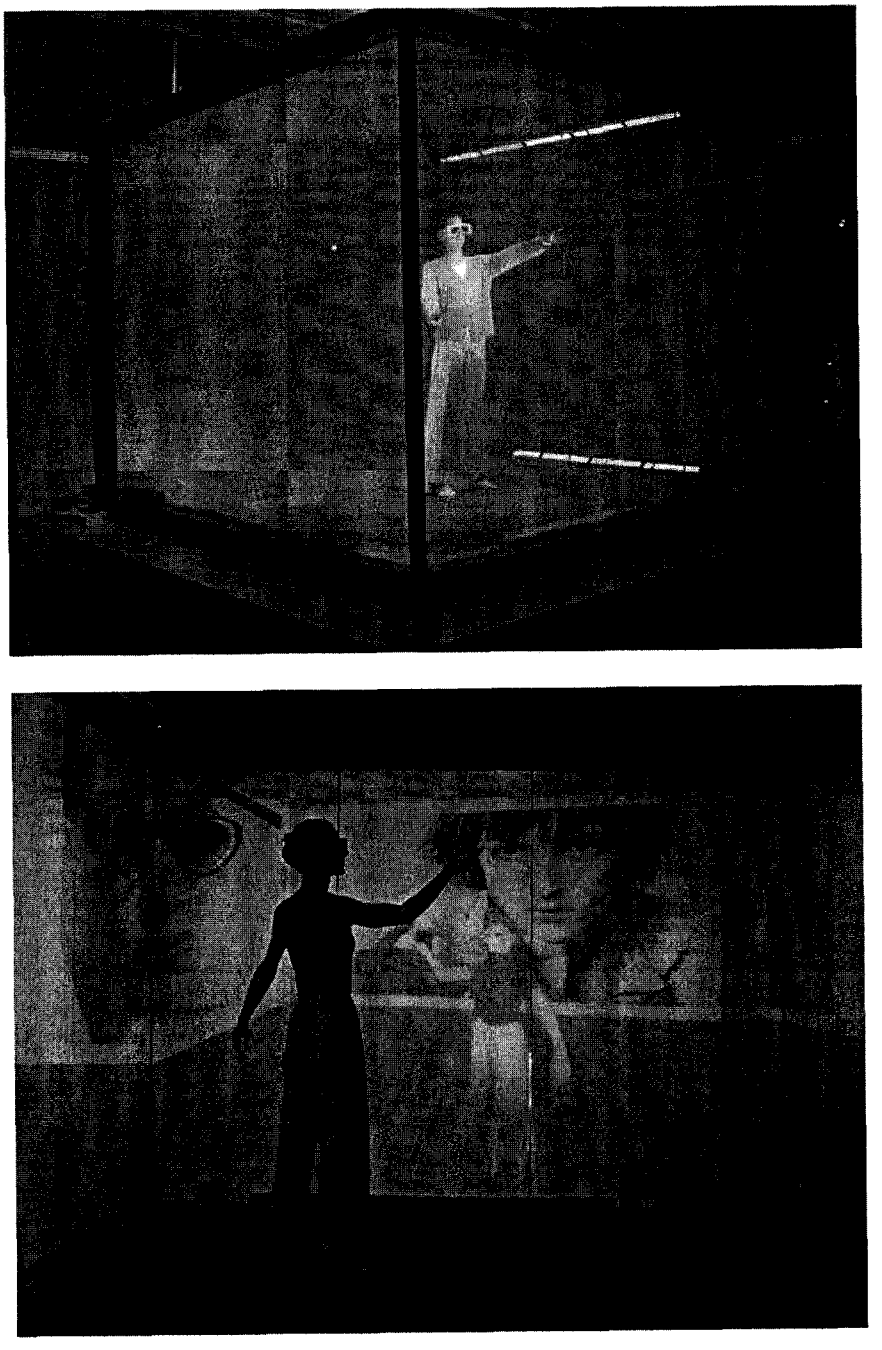


Figure 5: Blue-C: an immersive portal for tele-collaboration. Up: outside view while active screen is transparent. Down: inside view while active screen is opaque (shopping application) [59]

Recent research shows a trend that the realism of videoconferencing is embedded into shared virtual environments. Such systems not only can achieve realism and support physical presence but also perform interactions with objects of the virtual scene and do much more interesting things. It is believed that this evolution will end up with immersive portals that can fulfill the functionality of communication and collaboration. An innovative example is the blue-C [19]. It has a strong focus on the 3D video analysis and synthesis in a CAVE-like environment (Figure 5). Its work is close to NTII, but blue-C applies the new technology and uses a more sophisticated setup. It enables a number of participants to perceive photo-realistic 3D images of their remote partners in real-time, while interacting and collaborating inside an immersive virtual world. The key technology is a shuttered LCD screen that is switched from an opaque state for display (rear projection) to a transparent state for acquisition (cameras behind the screen). A special synchronization unit accurately switches between cameras, screen, projectors, stereo glasses and active illumination.

With the new 3D display technologies, users could perceive the immersive 3D scene without glasses, making the conferencing more natural. 3D TV system used rear-projection or front-projection lenticular screens (Figure6) to show high-resolution (1024×768) stereoscopic color images for multiple viewpoints without special glasses [20]. Although the overall delay (from acquisition to display) was less than one second, the system could not be directly used for teleconferencing because it could not afford to transmit 16-channel multiview video over peer-to-peer connections with current networking technology. Thereby, more efficient compression techniques are expected.

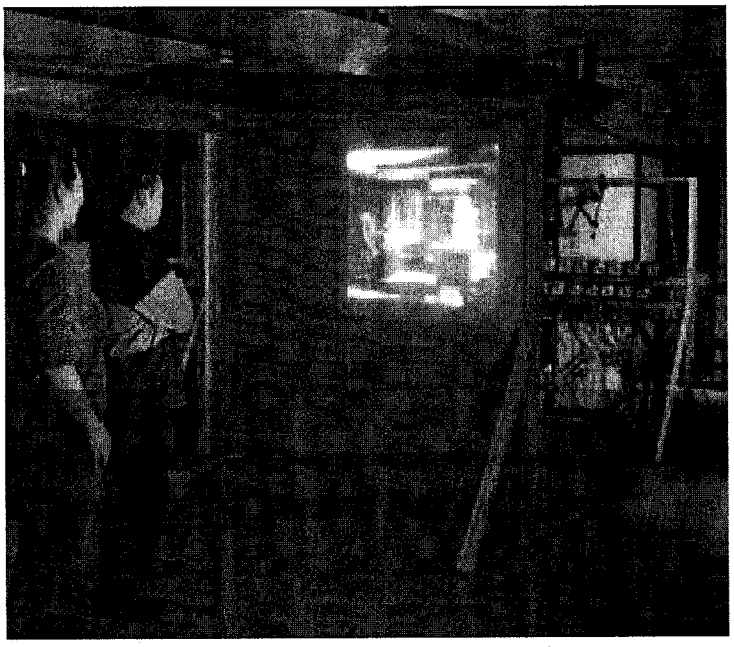


Figure 6: 3D TV system [20]

2.2 Geometry

This section introduces the basic concepts of geometry which are the theoretical foundation for the system design and implementation.

2.2.1 Perspective Projection Geometry

Figure 7 shows a pinhole camera model. It is depicted by its optical center **C** (also known as the camera projection center) and the image plane. The distance from C to the image plane is the focal length f. The line from the camera center perpendicular to the image plane is called the principle axis or optical axis of the camera. There is a three dimensional coordinate system, called the standard coordinate system of the camera,

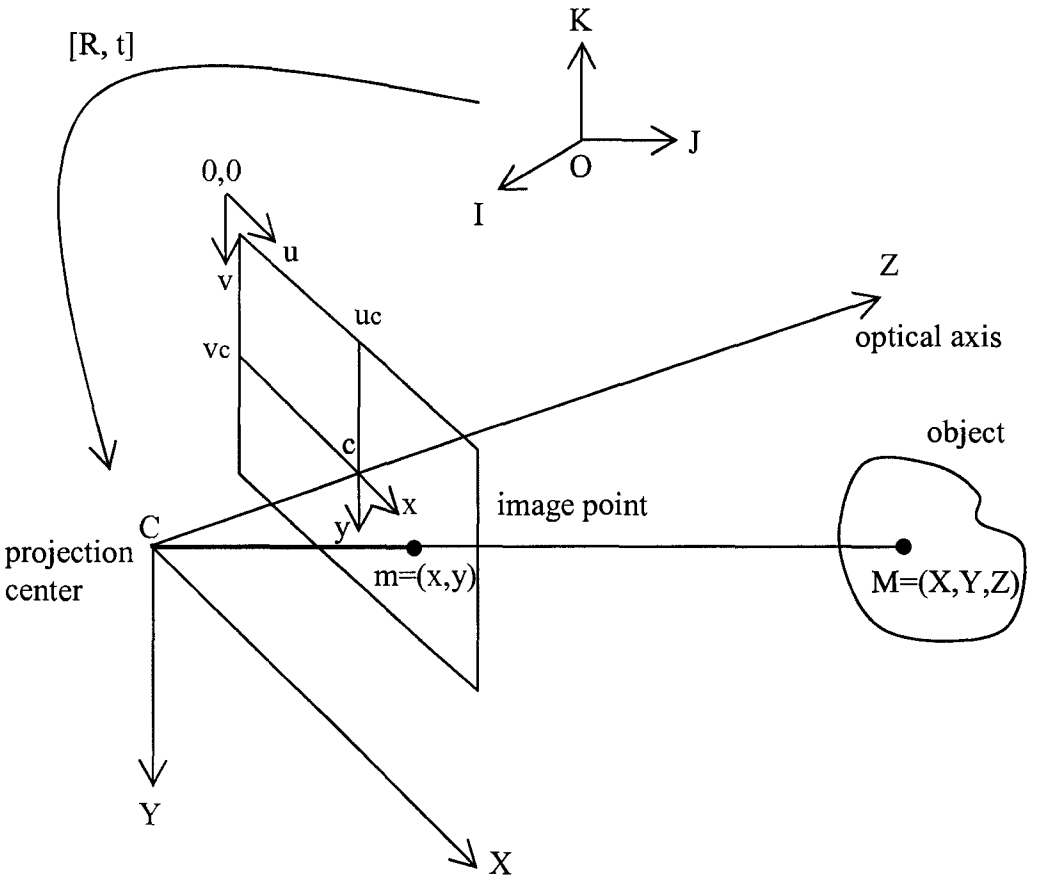


Figure 7: Perspective projection geometry

whose origin is at the center of projection and whose Z axis is along the optical axis. A point **M** on an object with coordinates (X, Y, Z) will be projected at some point $\mathbf{m} = (x, y)$ in the image plane. These coordinates are with respect to a coordinate system whose origin is at the intersection of the optical axis and the image plane, and whose x and y axes are parallel to the X and Y axes. The relationship between the two coordinate systems (C, X, Y, Z) and (c, x, y) is described by the perspective projection. By similar triangles the relationship is given by

$$x = \frac{Xf}{Z} \text{ and } y = \frac{Yf}{Z} \tag{2.1}$$

This can be written linearly in homogeneous coordinates as

$$\begin{bmatrix} sx \\ sy \\ s \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.2}$$

where $s \neq 0$ is a scale factor.

The origin of the actual pixel coordinate system (u, v) is defined at the top left corner of the image plane. The relationship between (x, y) and (u, v) is given by

$$u = u_c + \frac{x}{pixel\ width} \text{ and } v = v_c + \frac{y}{pixel\ height} \tag{2.3}$$

The transformation from three dimensional world coordinates to image pixel coordinates can be expressed by a 3×4 matrix. This is done by substituting equation (2.1) into equation (2.3) to obtain

$$Zu = Zu_c + \frac{Xf}{pixel\ width} \text{ and } Zv = Zv_c + \frac{Yf}{pixel\ height} \tag{2.4}$$

The perspective projection can be finally written in terms of matrix multiplication as

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} \frac{f}{pixel\ width} & 0 & u_c & 0 \\ 0 & \frac{f}{pixel\ height} & v_c & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.5}$$

where the scaling factor s has value Z. In short form is

$$\tilde{\mathbf{u}} = K \cdot \tilde{\mathbf{M}} \tag{2.6}$$

where $\tilde{\mathbf{u}}$ represents the homogeneous vector of image pixel coordinates, $\tilde{\mathbf{M}}$ is the homogeneous vector of world coordinates. K is the camera calibration matrix; it depends

on the so-called intrinsic parameters, i.e., focal length f, pixel width, pixel height, image center coordinates in pixels u_c, v_c.

In general, the three dimensional world coordinates of a point will not be specified in a coordinate system whose origin is at the center of projection and whose Z axis lies along the optical axis. Some other, more common frame (OIJK) will more likely be specified, and then a change of coordinates has to be performed from the general frame to the standard coordinate system. Thus

$$\tilde{\mathbf{u}} = K \cdot T \cdot \tilde{\mathbf{M}} = P \cdot \tilde{\mathbf{M}} \tag{2.7}$$

where T is a 4×4 homogeneous transformation matrix:

$$T = \begin{bmatrix} R & t \\ 0_3^T & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.8}$$

The top 3×3 corner is a rotation matrix R and encodes the camera orientation with respect to a given world frame; the final column is a homogeneous vector t capturing the camera displacement from the world frame origin. These parameters are known as the extrinsic camera parameters.

The camera calibration matrix K and the homogeneous transform T combine to form a single matrix P, called the camera projection matrix. It encodes the transformation from general world coordinates to image pixel coordinates.

2.2.2 Epipolar Geometry

The epipolar geometry describes the geometric relationship between two perspective views of the same 3D scene. The core idea is that corresponding image points must lie on particular image lines, which can be computed without information on the calibration of the cameras. This implies that, "given a point in one image, one can search the corresponding point in the other along a line and not in a 2D region, a significant reduction in complexity" [23]. CVonline [23] discusses the epiploar geometry in detail. Some definitions are given here.

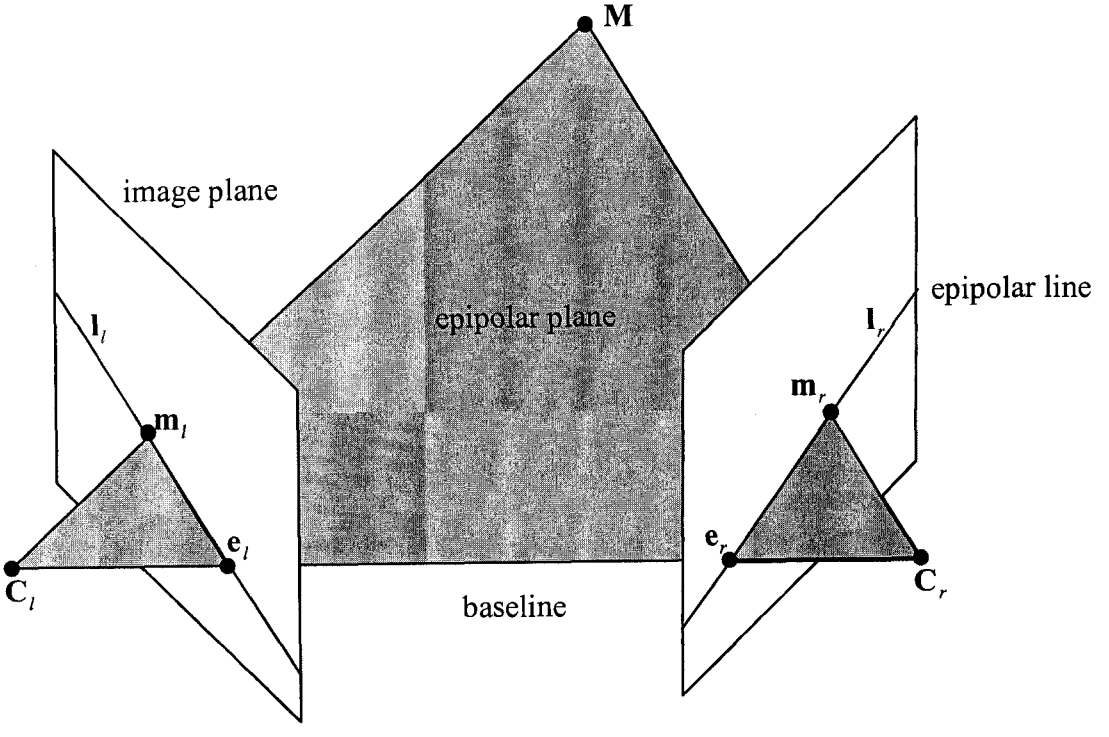


Figure 8: Epipolar geometry

Figure 8 illustrates the definitions of the epipolar geometry. Any 3D point **M** and the camera projection centers $\mathbf{C}_l$ and $\mathbf{C}_r$ define a plane that is called epipolar plane. The projections of the point **M**, image points $\mathbf{m}_l$ and $\mathbf{m}_r$, also lie in the epipolar plane since they lie on the rays connecting the corresponding camera projection center and point **M**. The conjugate epipolar lines, $\mathbf{l}_l$ and $\mathbf{l}_r$, are the intersections of the epipolar plane with the image planes. The line connecting the camera projection centers ($\mathbf{C}_l$, $\mathbf{C}_r$) is called the baseline. The baseline intersects each image plane in a point called epipole. By construction, the left epipole $\mathbf{e}_l$ is the image of the right camera projection center $\mathbf{C}_r$ in the left image plane. Similarly, the right epipole $\mathbf{e}_r$ is the image of the left camera projection center $\mathbf{C}_l$ in the right image plane. As the position of the 3D point **M** varies, the epipolar planes "rotate" about the baseline. This family of planes is known as an epipolar pencil. All epipolar lines in the left image go through $\mathbf{e}_l$ and all epipolar lines in the right image go through $\mathbf{e}_r$.

Chapter 3

Technologies

Many advanced technologies have been used to realize a major step towards a 3D immersive teleconferencing system, which include image segmentation, reconstruction, texture mapping, 3D display, gigabit network with real time transport protocol. They will be addressed as following sections.

3.1 Image Segmentation

3D reconstruction needs the silhouettes as input. Regions of interest in the images, i.e. the foreground or silhouette, are extracted by the processing of image segmentation. To distinguish between fore- and background pixels from image data, segmentation must be realized on the basis of the pixels' color. There are two well-known techniques for segmenting objects from the static background: chroma-keying and background subtraction.

3.1.1 Chroma-keying

Chroma-keying is a traditional method which is widely used in the TV and film industry. The foreground is recorded in front of a uniformly colored backdrop (typically blue or green) whose color does not resemble any color occurring on the foreground object. By labeling as background all pixels that are similar in color to the backdrop, the foreground can be quickly and robustly segmented. This method can be simply used in the analog field by constructing a band-pass filter. But in the digital domain, such a simple process cannot be as directly implemented, because most digital images are represented as a combination of red, green and blue components. First of all, the RGB values have to be converted to the HSV, where H (hue) is more or less an angle on a circle representing various chromatic values, S (saturation) is how much of that color is present in the pixel and V (value) is the pixel's "distance" away from black (see figure 9). Then in HSV space, we pick an angle corresponding to a range of color to be keyed, and pass all pixels that fall outside that value.

There are two problems using the approach of chroma keying. One is color bleeding, which means indirect illumination causes the background color of the walls and floor to reflect off the foreground object, resulting in segmentation inaccuracies and unnatural color tint on the object's texture. The other is shadows cast by the foreground onto the background, which also cause segmentation error. By changing light and material of backdrop, both problems can be solved.

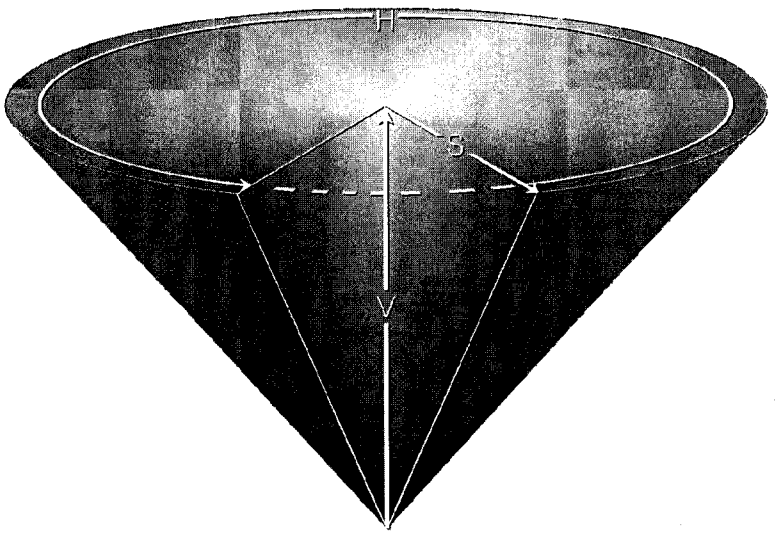


Figure 9: HSV cone [60]

For the purpose of making the system fast and robust, chroma-keying technique is used.

3.1.2 Background Subtraction

If the background cannot, or is not supposed to, be manipulated, segmentation is based on comparing the images with foreground to previously acquired images of the background. The region is identified as the foreground where the images differ; the region is identified as the background where they are the same. The amount of color bleeding from a typical background to the object is usually insignificant. However, the technique might identify some parts of the foreground as the background. This happens when the color of some portion of the object is very similar to the color of the background at that place. The technique might also identify some portions of the background as the foreground. This typically happens when the foreground object casts shadows onto the background. Only

comparing RGB values for each pixel between images is not enough. A lot of image processing techniques are used to deal with noise and shadows [24] [25] [26] [27].

3.2 3D Reconstruction

3D reconstruction is construction of the position and shape of objects from several two-dimensional views. There are basically two reconstruction approaches: depth-from-stereo and shape-based methods. If the input images are recorded from nearby viewpoints such that they exhibit high similarity, depth-from stereo algorithms that aim at assigning a depth value to each image pixel are applicable. For images captured from widely spaced positions, shape-based techniques which recover a full 3D model can be applied. In the following, both approaches will be introduced. One of the shape-based methods, Shape-from-silhouette which is used in my application will be described in more detail.

3.2.1 Depth from Stereo

Depth-from-stereo reconstruction from two or more images has been at the core of computer vision research for decades. For metric reconstruction, the recording cameras must be calibrated so that the images can be related geometrically. Figure 8 illustrates the epipolar geometry for a pair of stereo cameras. A 3D scene point M visible in both image $\mathbf{I}_l$ and $\mathbf{I}_r$ establishes a correspondence between two pixels $\mathbf{m}_l$ and $\mathbf{m}_r$ from either image. In reverse, a correspondence between two pixels determines the 3D position of the scene point. As we learned from Section 2.2, the search for correspondence can be constrained

to a 1D search along the epipolar line.

The core idea to solve the correspondence problem is similarity. Single color samples do not provide sufficient and distinct information for the correspondence search, hence similar areas or features are required in the two stereo images. Four approaches for correspondence search are commonly used and are reviewed in [56].

"Area-based algorithms consider an area or window around the pixel, in order to handle ambiguities in matching. They result in dense disparity maps, but fail in occluded or low-structure regions. Due to the classical depth reliability versus depth accuracy trade-off the definition of the window size is a key problem in area-based stereo [27]. As similarity measures, the sum of absolute difference (SAD), the sum of squared difference (SSD), or the normalized cross-correlation (NCC) [28] are commonly used.

Feature-based algorithms extract and match local feature like edge, corners or lines [29]. They provide sparse, but robust disparity maps.

Pixel-recursive approaches were originally developed for image sequence coding [30]. They obtain a dense field by scanning the whole image, predicting disparities from the previous one.

Optical-flow-based algorithms rely on the relation between photometric correspondence vectors and spatiotemporal derivatives of luminance [31] [32]." [56]

Depth-from-stereo approach requires a large amount of sufficiently proximal images to achieve effective correspondence, making it too slow for real-time use. Correspondence search is not robust especially in image regions with low or repeated texture, half-occluded regions and at occlusion boundaries. These disadvantages have led to an interest in shape-based methods.

3.2.2 Shape from Silhouette

To reconstruct 3D geometry from wide-baseline images, most time-critical systems rely on the shape-from-silhouette approach. Shape-from-silhouette is a class of algorithms for 3D scene reconstruction that uses the outline of the objects to recover their shape. These algorithms make use of the very efficiently reconstructible, yet only approximate, visual hull representation of actual object geometry.

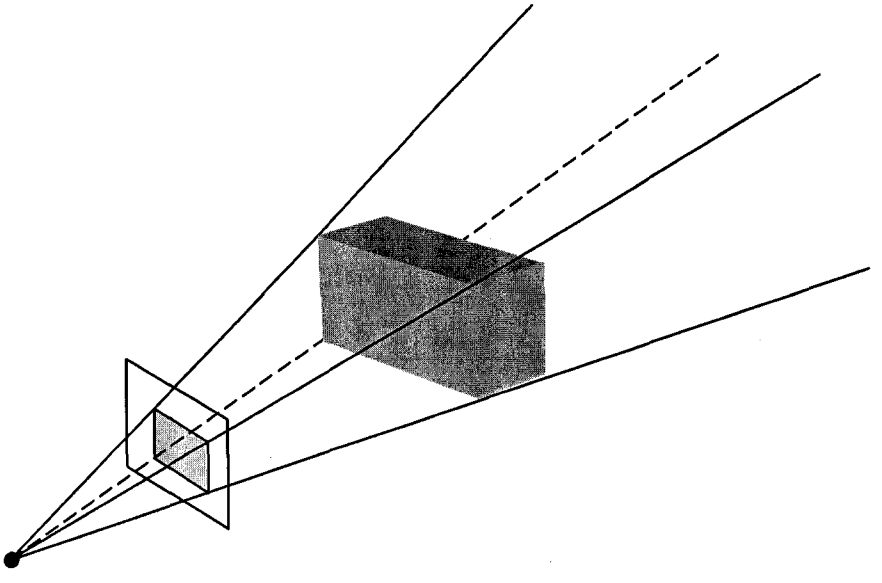


Figure 10: Viewing cone through the image silhouette containing the object

The basic idea of these approaches is that any object must be entirely located somewhere within its contour. If an object is shot by a pinhole camera, the rays starting from the projection center extending infinitely while passing through the silhouette on the image plane form a viewing cone. Although the actual direction of the rays is opposite, it does not affect the accuracy of these approaches. This viewing cone, illustrated in Figure 10, defines an upper bound for the object shape. The correct object volume is definitely less or equal than this rough approximation. For any camera, the silhouette defines a viewing cone which entirely contains the object. Since the object volume is bounded by all the viewing cones, it must also reside within the intersection of these viewing cones. Only points in the 3D space that are inside all viewing cones may belong to the object to be reconstructed. Figure 11 shows a planform of this process.

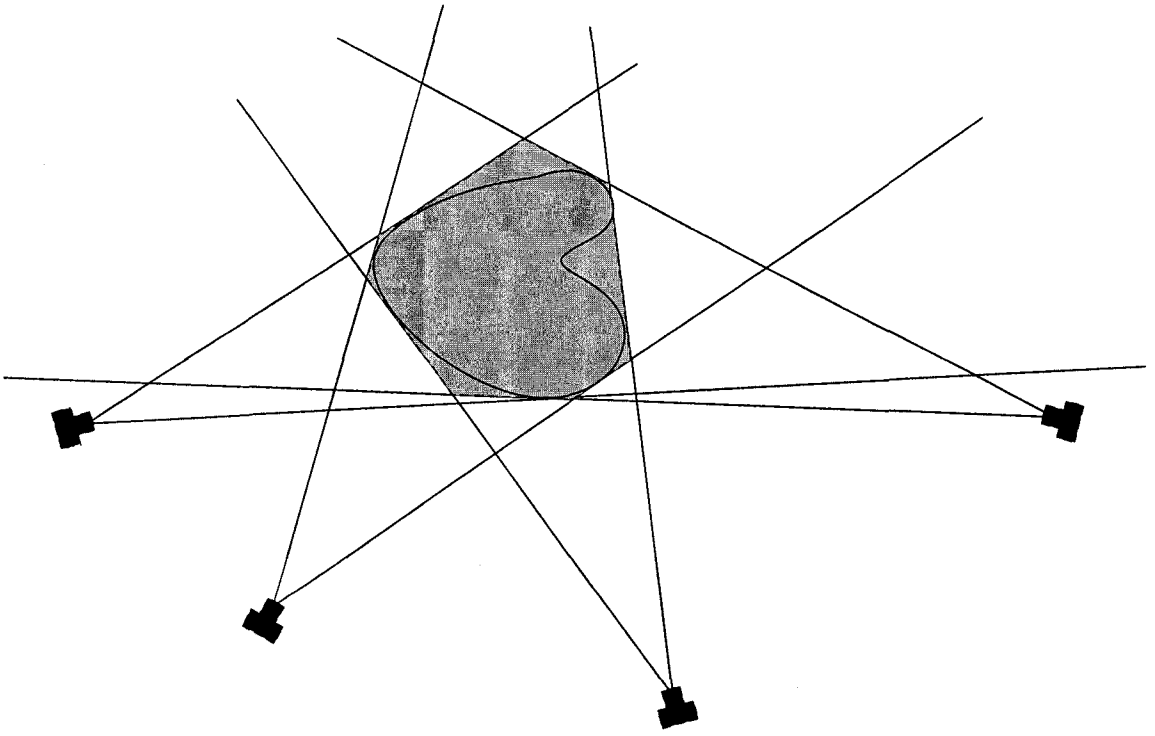


Figure 11: Intersection of four viewing cones

In theory, the visual hull is defined as the intersection of the cones from the infinite number of viewpoints surrounding the object [33]. This defines a maximal volume which is guaranteed to contain the object. In practice, it is infeasible to compute an infinite number of cones from which to recover the visual hull perfectly. So there is a tradeoff between speed and accuracy. Another limitation of shape-from-silhouette is its inability to recover concavities in surfaces (see Figure 11).

The most straightforward approach to compute visual hulls is the volumetric visual hull. The scene volume is divided into small cubic volume elements (voxels). Each voxel is projected into all source image planes and checked if it lies within the object's silhouette. Only voxels that consistently fall within the silhouettes in all images belong to the visual hull. Octree-hierarchies are often used to accelerate this procedure [34] [35].

Volumetric visual hull reconstruction does not provide a triangle mesh of the visual hull surface that could be rendered right away by hardware. Alternatively, a polyhedral approach towards real-time visual hull rendering can be pursued by describing silhouette contours as polygons [36]. This algorithm will be described in 3.2.3.

If the primary purpose of a shape representation is to produce new rendering of that shape from different viewing conditions, then construction of an explicit model is not necessary. The image based visual hull (IBVH) technique [37] renders unique views of the visual hull directly without constructing an intermediate volumetric or polyhedral model. This is accomplished by combining the cone intersection calculation with the rendering process.

The rendering process which could be done by a graphics card is performed by software, making IBVH slow.

3.2.3 Polyhedral Visual Hull

Polyhedral Visual Hull is an efficient algorithm to compute the intersection of the viewing cones. In order to compute the visual hull with respect to the input silhouettes, the intersection of the viewing cones needs to be computed. The resulting polyhedron is described by all of its faces. The faces of this polyhedron can only lie on the faces of the original viewing cones. The faces of the viewing cones are defined by the projection matrices and the edges in the input silhouettes. The simple PVH algorithm is as follows: First, smooth silhouette contours are converted into polygons. Second, "for each input silhouette s_i and for each edge e in the input silhouette s_i, the face of the cone is computed. Third, the intersection of this face with the cones of all other silhouettes is performed. The result of these intersections is a set of polygons that define the surface of the visual hull [36]".

As the calculation of the intersection of polyhedral cones is in 3D and thus computationally expensive, the algorithm proposes to reduce the calculation into 2D by employing features of epipolar geometry. This is because each of the silhouette cones has a fixed scaled cross-section. Thus intersection in 2D can be achieved as follows. "First project a cone face f of a cone c_i onto the image plane of silhouette s_j (see Figure 12). Then intersect projected face f_p with silhouette s_j. Finally, project back the resulting

polygons onto the plane of face f [36]". After repeating this operation for every projected cone face in c_i, the entire viewing cone c_i is intersected with silhouette s_j.

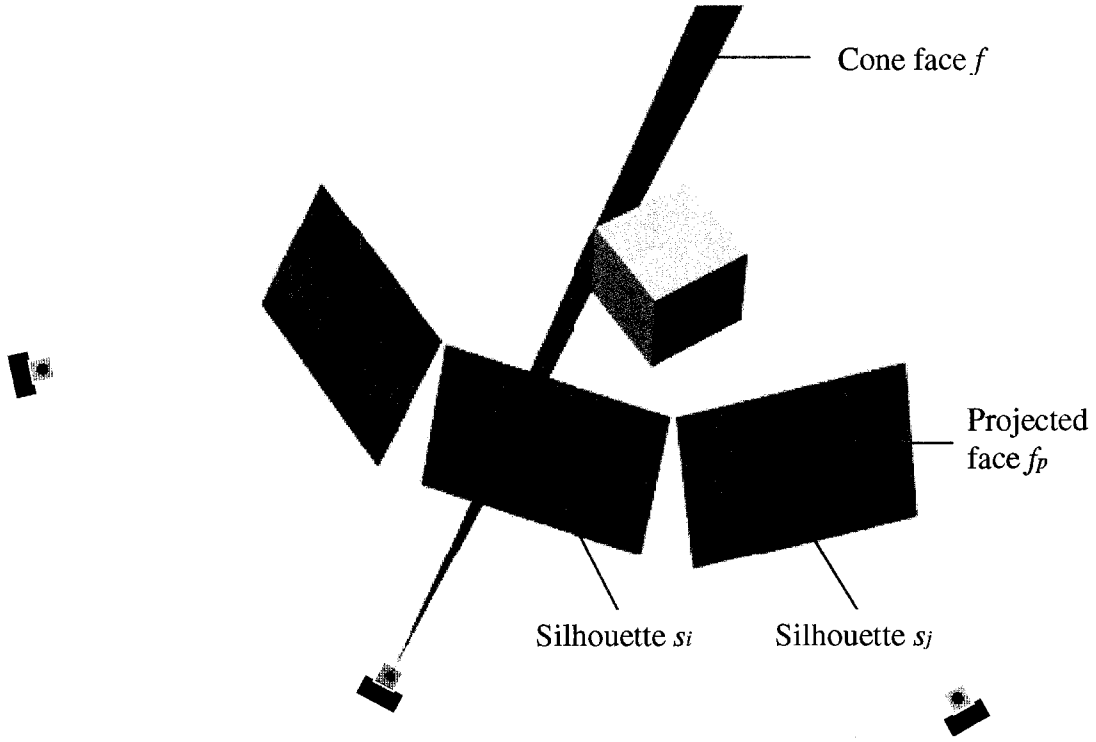
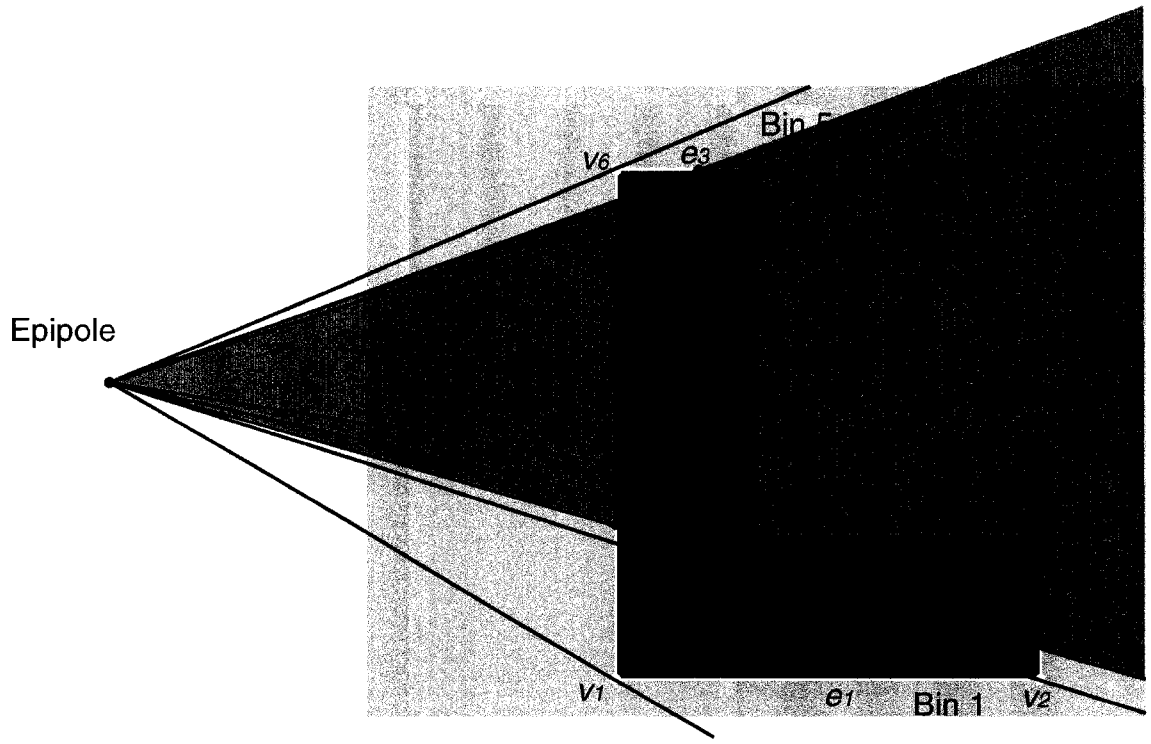


Figure 12: A single silhouette cone face and its projection in two other silhouettes [36]

To efficiently compute the intersection of projected face f_p with silhouette s_j, Edge-Bin data structure is applied. The Edge-Bin structure spatially partitions a silhouette so that the set of edges that a projected cone face intersects can be computed quickly.



Bin b_k	1	2	3	4	5
Edges S_k	e_2, e_1	e_2, e_6	e_2, e_4, e_5, e_6	e_2, e_4	e_2, e_3

Figure 13: Intersection of a projected face f_p of viewing cone c_i with silhouette s_j and Edge-Bin data structure

According to epipolar geometry, all rays on the surface of the cone c_i project to a pencil of lines sharing a common point p_0 (i.e., the epipole) in the image plane of s_j. All projected lines are parameterized based on the slope α that these lines make with some reference line. A bin b_k is defined to be a three-tuple: the start α_{start}, the end α_{end} of the range, and a corresponding set of edges S_k, $b_k = (\alpha_{start}, \alpha_{end}, S_k)$. As shown in Figure 13, each silhouette vertex corresponds to a line that defines an area boundary. The Edge-Bin construction includes two steps. First, the silhouette vertices are sorted in the order of

increasing α. The lines that pass through the silhouette vertices define the bin boundaries. Two consecutive slopes in the sorted list define α_{start} and α_{end} for each bin. Next, to compute a set of edges assigned to each bin, the sorted list of silhouette vertices is traversed. At the same time the list of edges in the current bin is maintained. When a vertex of the silhouette is visited, an edge that ends at this vertex is removed from the current bin and an edge that starts at the vertex is added. The start point of an edge is defined as the point that has a smaller value of parameter α. The algorithm of step two can be written as follows:

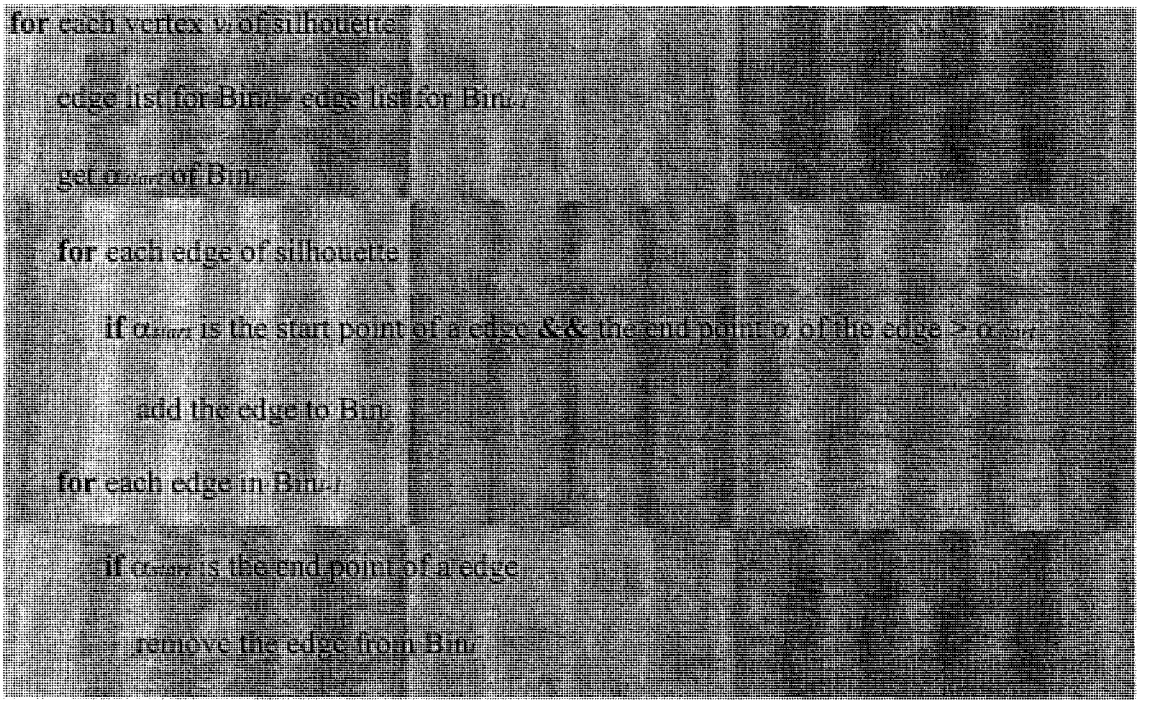


Figure 13 shows an example of a silhouette, bins, and corresponding edges for each bin. Vertices are traversed in the direction of increasing α (from v_1 to v_6). For bin 1, its lower boundary α_{start} is the α corresponding to vertex v_1. At this vertex, two edges begin (e_2 and e_1) so both are added to the bin's list of edges. For bin 2, its lower boundary α_{start} is the α corresponding to vertex v_2. *v_2 is the start point of e_6* and the end point of e_1, so add e_6

and remove e_1. For bin 3, its lower boundary α_{start} is the α corresponding to vertex v_3. *v_3 is the start point of* e_4 and e_5, so add e_4 and e_5. For bin 4, its lower boundary α_{start} is the α corresponding to vertex v_4. *v_4 is the end point of* e_5 and e_6, so remove e_5 and e_6. For bin 5, its lower boundary α_{start} is the α corresponding to vertex v_5. *v_5 is the start point of* e_3 and the end point of e_4, so add e_3 and remove e_4.

After constructing the Edge-Bin data structure, the following task is to compute the intersection of the first projected face f_{p1} with the silhouette s_j. f_{p1} is defined by its boundary lines with the values α_{p1}, α_{p2}. First, find a bin $b = \{\alpha_{start}, \alpha_{end}, S\}$ such that $\alpha_{p1} \in (\alpha_{start}, \alpha_{end})$. Then, intersect the line α_{p1} with all the edges in S. Next, traverse the bins in the direction of the value α_{p2}. While moving across the bins, the intersection polygons are built by adding the vertices that define the bins. When getting to the bin $b' = \{\alpha'_{start}, \alpha'_{end}, S'\}$ such that $\alpha_{p2} \in (\alpha'_{start}, \alpha'_{end})$, the line α_{p2} is intersected with all edges in S' and the remaining edges of the resulting polygons are computed. Because the faces of cone c_i are processed in consecutive order, the next projected face f_{p2} is defined by the boundary lines α_{p2}, α_{p3}. Therefore, there is no need to search for the bin α_{p2} falls into again. In this manner the intersection of all projected faces of cone c_i with the silhouette s_j is computed. Figure 13 shows an example. Blue area is a projected face. The area surrounded by red lines is the intersection of a projected face with a silhouette.

"After performing the pair wise intersection on all pairs of cones, $k - 1$ polygon sets are computed for each face of each cone. Here k is the total number of silhouettes. The faces of the visual hull are the intersections of these polygon sets at each cone face. PVH

polygon intersection routine separates polygons into quadrilaterals and intersects those. Figure 14 demonstrates the procedure with two 5-sided polygons, one with vertical hatching and the other with horizontal hatching. First the polygons are divided into quadrilaterals based on the polygons' vertices and the apex of the silhouette cone (similar to the Edge-Bin construction process). The quadrilaterals are the parts of the triangles that shared the same vertex (i.e., the projection center). Then in each triangular region quadrilaterals are intersected. Last all of the intersection results are combined into the final polygon, shown with both horizontal and vertical hatching [36]".

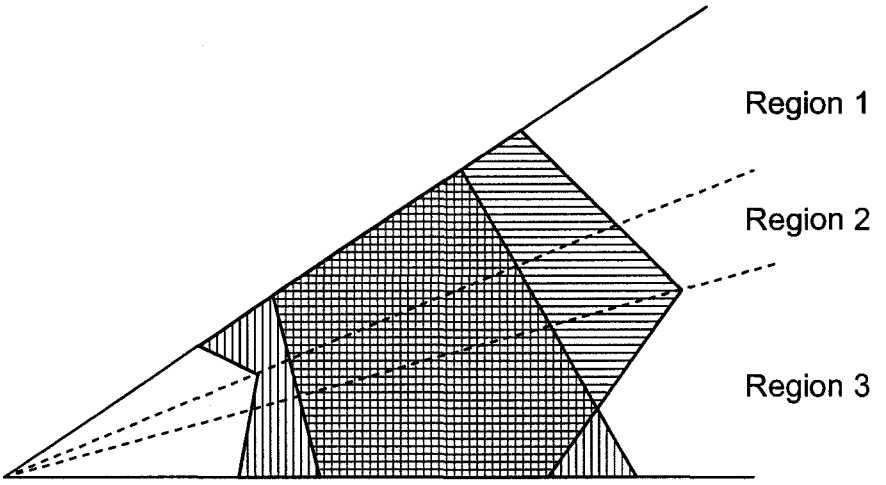


Figure 14: Polygon is subdivided into quadrilaterals for intersection

3.2.4 Exact Polyhedral Visual Hull

EPVH is more efficient than PVH. PVH approach duplicates cone intersection operations and requires an additional step to connect the different parts of the visual hull, with no

topological guarantee. The EPVH algorithm recovers the complete visual hull polyhedron as a whole with a reduced number of operations.

EPVH is more robust than PVH. Because of PVH non-optimal face intersection routine, meshes may contain T-junctions which violate the watertight property. Therefore PVH may produce a non-watertight triangular mesh. EPVH recovers a both exact and watertight visual hull polyhedron.

The EPVH algorithm involves three main steps. "First, a coarse geometrical approximation of the visual hull is computed by retrieving its viewing edges, an unconnected subset of the wanted mesh. Then, local orientation and connectivity rules are used to walk along the relevant viewing cone intersection boundaries, so as to iteratively generate the missing surface points and connections. A final connection walkthrough allows identifying the planar contours for each face of the polyhedron." The details about this algorithm are in [38].

3.3 Texture Mapping

Texture mapping has been widely applied in computer graphics and virtual reality. Without it, modeling and rendering the details of complex models would be a time-consuming task. The fundamental issue of texture mapping is to construct a one-to-one mapping between each point on the specified surface area of a 3D object and that on the texture plane (see Figure 15).

The surface of a model is composed of triangles. The most important thing is for each triangle in the model to find a corresponding region in the texture image. In this section I first describe how to project a single image onto the model, and then how to merge several image projections to render the entire model.

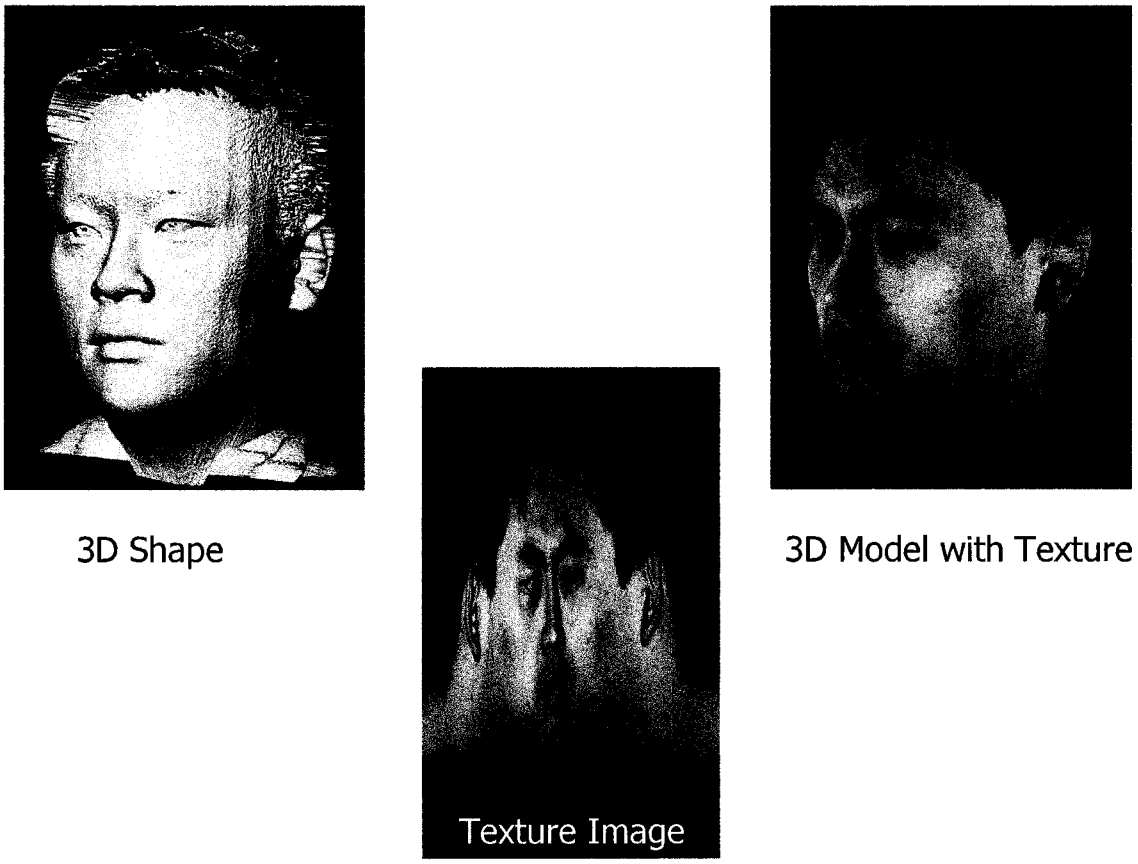


Figure 15: Texture mapping

3.3.1 Projective Texture Mapping

Projective texture mapping was introduced in [39] and is now part of the OpenGL graphics standard. Although the original paper used it only for shadows and lighting

effects, it is directly applicable to image-based rendering because it can simulate the inverse projection of taking photographs with a camera. In order to perform projective texture mapping, the user specifies a virtual camera position, and a virtual image plane with the texture. The texture is then cast onto a geometric model using the camera position as the center of projection. Since the camera positions and orientations are given by camera calibration, projecting the images onto the model is a straightforward way to fulfill the texture mapping. In other words, Camera calibration told us the relationship between the vertices in the mesh and those in the texture image plane (equation 2.7). As shown in Figure 16, for each triangle in the mesh find the corresponding triangle in the texture image, and then simply project back that texture to the mesh.

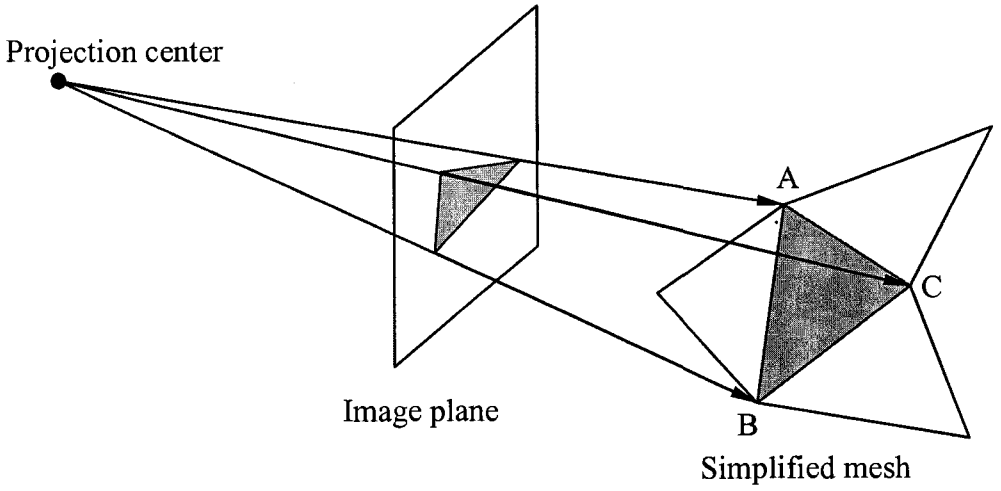


Figure 16: Projective texture mapping

3.3.2 View-dependent Texture Mapping

The traditional projective texture mapping use only one camera's image as texture. So some faces of the model that can not be seen by that camera will get wrong texture. View dependent texture mapping method projects different images onto the model depending on the user's viewpoint. The basic approach to view dependent texture mapping (VDTM) is put forth by Debevec et al. [40] [41] in their Facade image-based modeling and rendering system. Facade is designed to estimate geometric models consistent with a small set of source images. As part of this system, a rendering algorithm was developed where pixels from all relevant cameras were combined and weighted to determine a view dependent texture for the derived geometric models.

In my application, the 3D geometry model of the scene is available after reconstruction. Novel views can be rendered in real time using view-dependent texture mapping. The geometry proxy is a triangular mesh representing (approximate) 3D scene geometry, while the calibrated input images serves as texture.

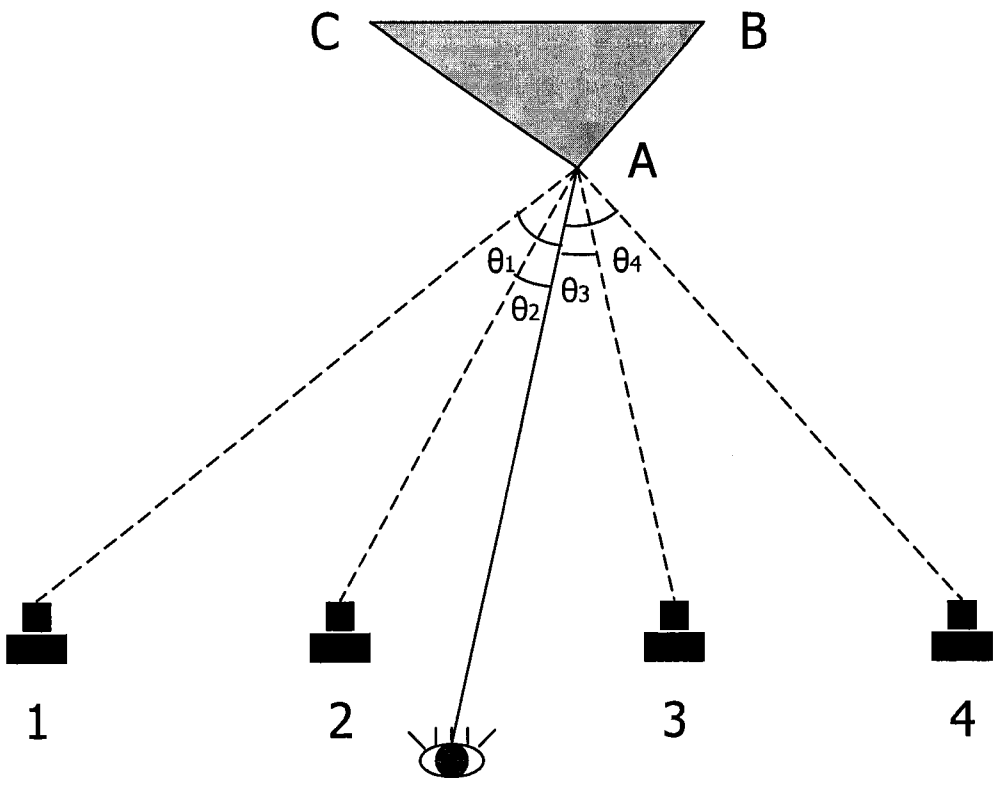
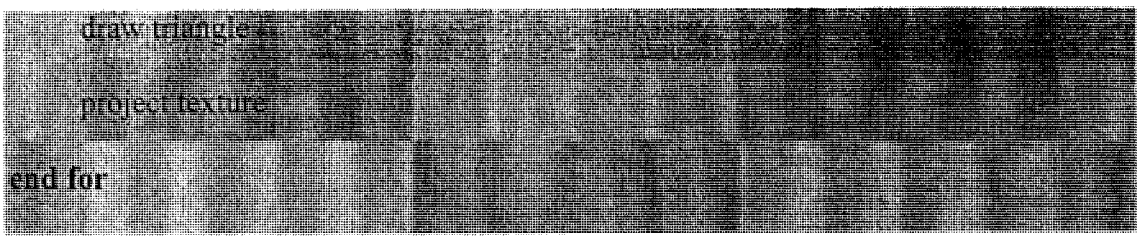


Figure 17: View-dependent texture mapping

My simple pseudocode for view dependent texturing as follows:

```
specify a viewpoint
for each points of mesh do
    calculate the angles between camera viewing rays and desired viewing ray
    find the smallest angle
    set a camera image for texturing
end for
for each triangle do
    if more than one vertex choose the same camera image for texturing
        set the camera image for texturing
```



For each triangle, choose the closest camera to the viewpoint for texturing. Take triangle ABC as an example (see Figure 17), θ is the angle between the desired viewing ray and the camera viewing rays. For vertex A, θ_2 is the smallest angle. Camera 2 is therefore chosen. If vertex B or C also choose camera 2, we use that camera 2's image is used for texturing.

There are two problems. One is that if the three vertices of a triangle choose the different cameras, how to texture the triangle? I solve this problem by just specifying a camera for that triangle. It's not a big problem. Compared with distance between a model and a camera, the triangle size is very small. The three vertices of a triangle always choose the same camera. This problem happens scarcely.

The other problem is that texture does not change smoothly when viewpoint change in the middle of two cameras. To avoid this problem, a weighting function is used. The pixel in the desired view corresponding to the point on the model is assigned a weighted average of the corresponding pixels in camera views. The weights which are inversely proportional to the magnitude of angles are stored in the alpha channel.

The popular weighting is the k-nearest neighbor weighting used in [42] and summarized here. For each vertex of the model, the k cameras are found whose viewings rays to that vertex are closest in angle to the desired viewing ray (see Figure 17). Consider the k th ray with the largest viewing angle, θ_k. This angle is used to define a weighting function that maps the other angles into the range from 0 to 1: $weight\ (\theta) = 1\text{-}\theta/\theta_k$. Applying this function to the k angles, results in $k-1$ non-zero weights. These weights are renormalized to get the final blending weights. To make computation fast, $k = 3$ is practically chosen in the four camera system, which results in two non-zero weights at each vertex.

3.4 3D Displays

Our visual system utilizes visual cues to depth to construct a perception of depth. Natural scenes contain a wide variety of visual cues to depth. They are classified into two categories. One is called the monocular cues. The effectiveness of monocular cues is demonstrated by the fact that we still have a considerable appreciation of depth with only one eye. Monocular information such as accommodation, occlusion, relative size, linear and aerial perspective, relative density, and motion parallax is already available in monoscopic displays, such as traditional 2D television.

The other is named the binocular cues. Binocular depth cues, which include stereopsis and disparity, require both eyes to work in concert. This is illustrated by the fact that we close one eye and can not exactly touch a point located in the 3D world with a finger. Stereopsis is available because the human eyes are horizontally separated (on average 6.3

cm) which provides each eye with a unique view of the world. This horizontal separation causes an interocular difference in the projections of 3D scenes onto the left and right retina, i.e., binocular disparity. When points from one eye's view are matched to corresponding points in the other eye's view, the retinal point-to-point disparity variation across the image provides information about the relative distance of objects to the viewer and the depth structure of environments. Stereoscopic vision greatly enhances human ability to extract a fairly accurate estimate of depth, particularly at close distances.

In the immersive 3D virtual environment applications, 3D displays are required to enhance the spatial impression. There are basically three types of 3D displays [20] [43]: stereoscopic displays, volumetric displays and holographic displays. Stereoscopic displays use various methods to convey a separate image to each eye, allowing the perception of depth. Volumetric displays employ a medium to fill or scan a three-dimensional space to produce volume-filling images. 3D image points (voxels) are addressed and illuminated individually. Holographic displays create a light field which is identical to that which emanated from the original scene, by using coherent light to illuminate the interference pattern which records the light wavefront of the original object. The technique behind this is holography which can record and reproduce the properties of light waves - amplitude (luminance), wavelength (chroma) and phase difference (distance).

In the proposed 3D teleconferencing system, a mature and well-performed stereoscopic display, Polarization-multiplexed display is employed. It will be introduced in detail.

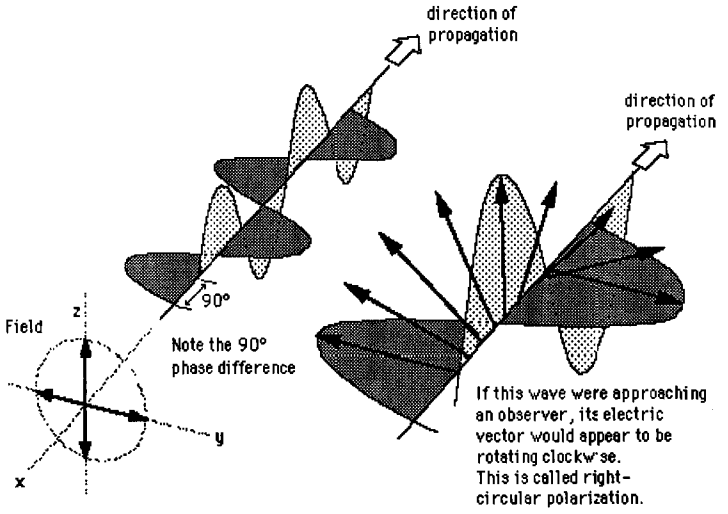


Figure 18: Linear and circular polarization [58]

3.4.1 Polarization-multiplexed Display

Polarization-multiplexed displays polarize the left and right eye images in orthogonal directions (linear or circular see Figure 18). At the DISCOVER lab, there is a circular polarization multiplexed display system named DIVINE (Desktop-Immersive Virtual and Interactive Networked Environment System). The basic setup, as shown in Figure19, consists of two LCD projectors arranged at a right-angle, circular polarization filter sheets in front of each projector and a non-depolarizing screen. The two views are combined by a beam combiner, first-surface mirror, which is frequently used in rear projection systems design to fold the light from the projector, reducing the required depth of the projection area behind the rear screen. Although over 60% of the light is lost through the filters, the first-surface mirror reflects 90% of the incident light, maintaining

enough light flux. The user wears polarized glasses. The polarizing lenses of the glasses merge the polarized light from the display device to act as blocking shutters to each eye.

This technique provides full color rendition at full resolution and very little cross-talk in the stereo pair (less than 0.1% with linear filters).

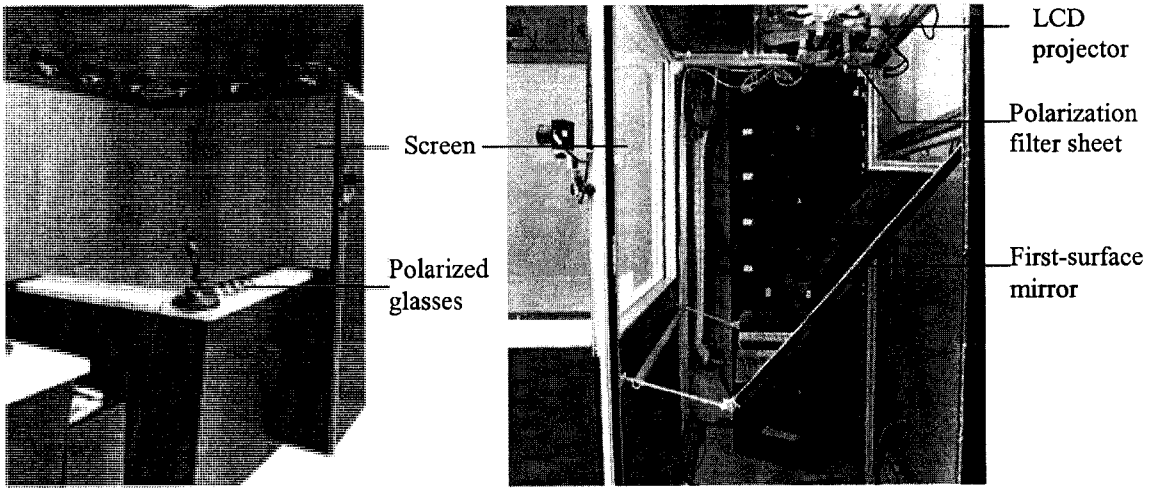


Figure 19: The DIVINE system at the DISCOVER lab

3.5 Real-time Transport Protocol

The Real-time Transport Protocol (RTP) provides end-to-end delivery services for data (such as interactive audio and video) with real-time characteristics. RTP was developed by the Audio/Video Transport working group of the IETF and has since been adopted by the ITU as part of its H.323 series of recommendations, and by various other standards organizations. The first version of RTP standard was completed in January 1996 [44]. More recent versions are RFC 3550 [45] and RFC 3551[46]. Although RTP was

originally designed for use in multicast conferences, it is used for different types of real time applications, such as H.323 video conferencing, webcasting, TV distribution and wired or cellular telephony.

RTP itself does not provide any mechanism to ensure timely delivery or provide other quality-of-service guarantees, but relies on lower-layer services to do so. "RTP is best viewed as a framework that applications can use directly to implement a single protocol. Without the application-specific information, RTP is not a full protocol. However RTP imposes a structure and defines common functions so that individual real-time applications are relieved of part of their burden" [62].

RTP is made up of two parts. One is the real-time transport protocol (RTP), to carry real-time data. The other is the RTP control protocol (RTCP), to monitor the quality of service and to convey information about the participants in an on-going session.

There are three reasons why the RTP is chosen for the presented 3D teleconferencing system. First, RTP is suitable for real-time communication, especially for video conferencing. The sequence numbers included in RTP header allow the receiver to detect packet loss and reconstruct the sender's packet sequence. Timestamp generated from a local clock at the source is good for synchronization between different media streams, such as graphics and texture. Second, RTP runs on top of UDP, so it is easy to implement. The third reason is that RTP is easy for multicast. In the system, mesh and texture are sent to three machines by multicast.

Chapter 4

Implementation

The proposed 3D teleconferencing system consists of several components, such as capture, segmentation, reconstruction and rendering. Each part performs a relatively independent operation. As main technologies used in each part have been introduced in chapter 3, this chapter focuses on many complex design considerations involving hardware component selection, software architecture, and physical layout of the acquisition and rendering environment. In the stages of the implementation, initially, each part is implemented and tested individually. The communication media between the different parts are files, such as image files (in bmp, tif or pgm formats) and mesh file off (details refer to [47]). Then, all the parts are integrated using network communication and memory operation. Last, the whole system is tested on-line with real-time generated dynamic models. For the network communication, RTPlib API [48] is employed to implement RTP.

4.1 System Overview

A typical video conferencing system is bilateral which allow two sites to communicate with each other over some distance. Primarily for economic reasons, the current system is asymmetric. The 3D video scene is reconstructed on the local site and displayed on the remote site. The system pipeline is shown in Figure 20, where two sites are connected through the network. At the local site, 4 calibrated Point Grey Research Dragonfly FireWire cameras are deployed, which are connected to a dedicated computer (equipped with 3.20GHz Pentium 4 CPU) through a hub. The cameras share with a single IEEE 1394 bus so that they are automatically synchronized to each other at the hardware level (the maximum deviation is 125μs). The computer captures the 640×480 video frames at up to 15 fps (30 fps exceeds the 400Mbit/s FireWire bandwidth) and performs the image segmentation. Because of its limited computation capability and the existing heavy processing load on it, the silhouettes are sent over a 1Gb/s Ethernet to another computer (with 2 3.06GHz Xeon CPU) to perform the 3D reconstruction, in a distributed computing fashion. Texture and mesh are sent to the remote site for rendering. At the remote site, there is a rendering cluster made up of 3 dual-processor computers (with 3.06GHz Xeon CPU). Each computer drives a 3D stereo display. The mesh and texture therefore are multicasted to the three machines. All in all, the proposed 3D teleconferencing system consists of four components involving capture, segmentation, reconstruction and rendering. The corresponding results after each step are shown in figure 20.

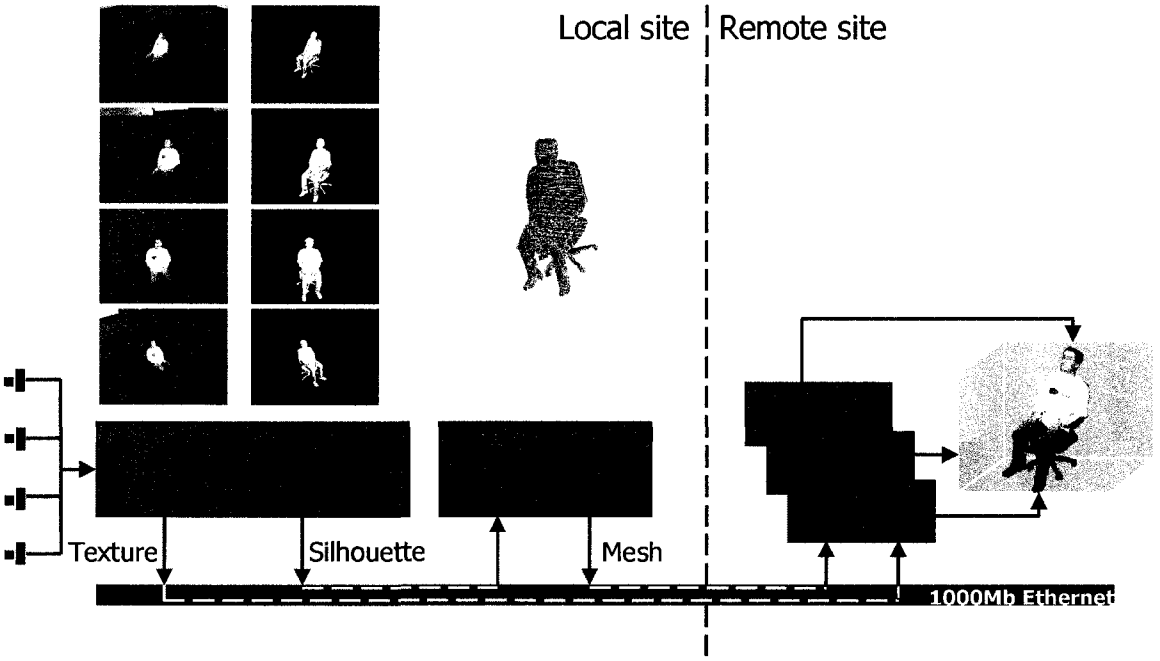


Figure 20: System pipeline

4.2 Prerequisite Work

4.2.1 Capture Environment Set-up

The 3D teleconferencing system currently operates in half-duplex mode, which means 3D acquisition and 3D display are not currently co-located. As such, camera layout is unconstrained by display placement. As shown in Figure 21, four cameras are mounted on a wall of the green room at the DISCOVER lab. Each camera is aimed at the conferee. The reason why no cameras are in the back of the conferee is that the view point is always in the front of the model. The cameras must be installed in a stable fashion so they do not accidentally change position or orientation. In order to perform the chroma keying,

a green screen is used as a backdrop. Considering the movement of the model, the green screen should be big enough to keep the background always green.

Figure 21: Capture environment set-up

For the purpose of duplex mode, cameras and 3D display need be co-located. As shown in Figure 19, four cameras are mounted on the 3D display, half surrounding the conferee.

4.2.2 Camera Calibration

The projection matrices which are required for the purposes of reconstruction and projective texture mapping are obtained from camera calibration. The calibration for 4 cameras is performed using GML C++ Camera Calibration Toolbox [49]. First, for each camera, multiple snapshots of a moving checkerboard pattern in various positions are taken. Figure 22 shows a snapshot taken from a camera. The checkerboard pattern has

11×10 squares. Each square size is 100mm×100mm. The pattern is so big that it can be clearly captured by all the four cameras simultaneously, making calibration accurate. Second, the calibration patterns in the images are detected automatically by the software tool, which is more convenient than Matlab calibration toolbox [50], because the users do not need to specify the four corners of the checkerboard. After the detection, the corners of the squares are marked by colorful points, as shown in Figure 22. Third, the cameras are calibrated to get the intrinsic parameters. It is important to note that the extrinsic parameters obtained here do not present the real camera's position and orientation. They just present the relative position and orientation to the pattern. Fourth, four pictures of the fixed pattern are taken simultaneously by the four cameras. Repeat step 2 and 3 to obtain the extrinsic parameters. The extrinsic parameters obtained present the real cameras' position and orientation in a fixed world coordinate system. As shown in Figure 22, the pattern's plane is the XY plane of the world coordinate system. One of the pattern's corners is the origin of the world coordinate system. Usually the origin of the world coordinate system is set at the location of one of the camera. So some transformations are performed for the new world coordinate system. Last, the camera projection matrices are computed using equation (2.7).

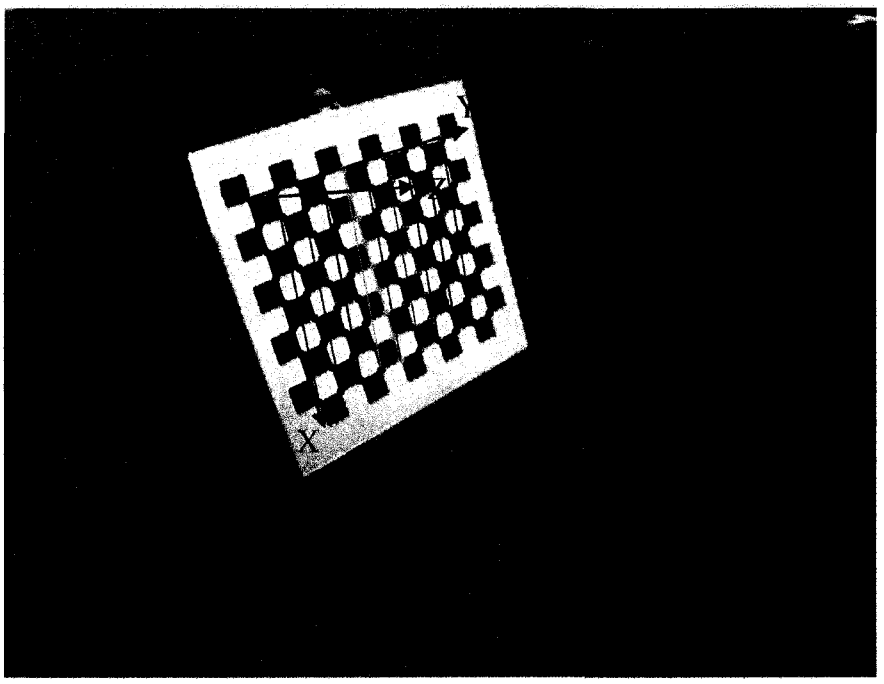
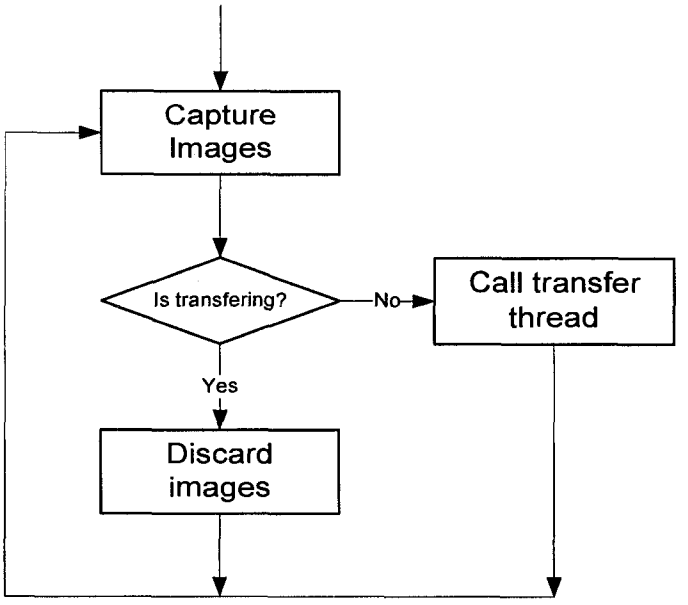


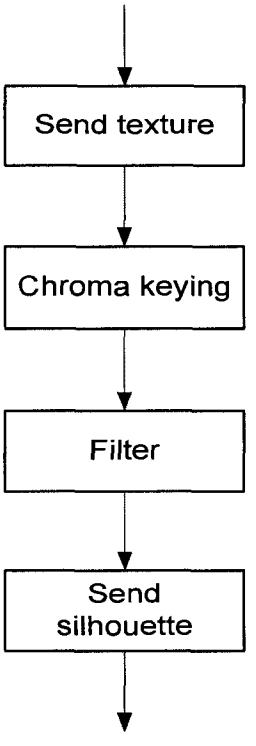
Figure 22: A snapshot of the checkerboard pattern from a camera and the world coordinate system

4.3 Capture

Capturing images from multiple cameras is performed by the PGR Flycapture API [61] which has a special buffer mechanism to guarantee that no images are missed and that all of the images are synchronized. The images grabbed are 8-bit per pixel raw images which have to be converted to 24-bit per pixel images, making further image processing possible. This process is performed by calling a function of the PGR Flycapture API.



Main process



Transfer thread

Figure 23: Flow chart of capture site

Figure 23 shows the flow chart of the capture site. The main process and the transfer thread run simultaneously. The main process captures images, while the transfer thread transmits texture images, performs chroma keying and transmits silhouettes. The two threads share the image buffer. The capture thread writes the image buffer, while the transfer thread reads the image buffer. The simultaneous reading and writing operations at the same memory location may cause conflicts. Once a group of four images is captured, the transfer thread is first checked whether or not it is in the processing of transmission of images. If so, the captured images are discarded. Although the action avoids conflicts between reading and writing, there are some side effects of this algorithm that will be discussed in section 5.2.

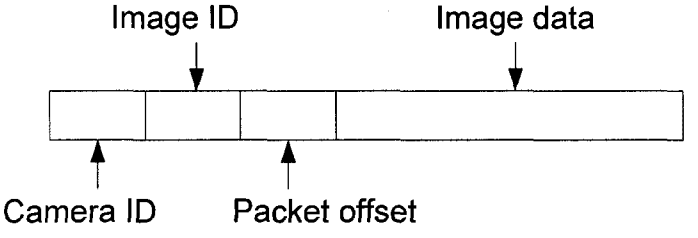


Figure 24: Image packet structure

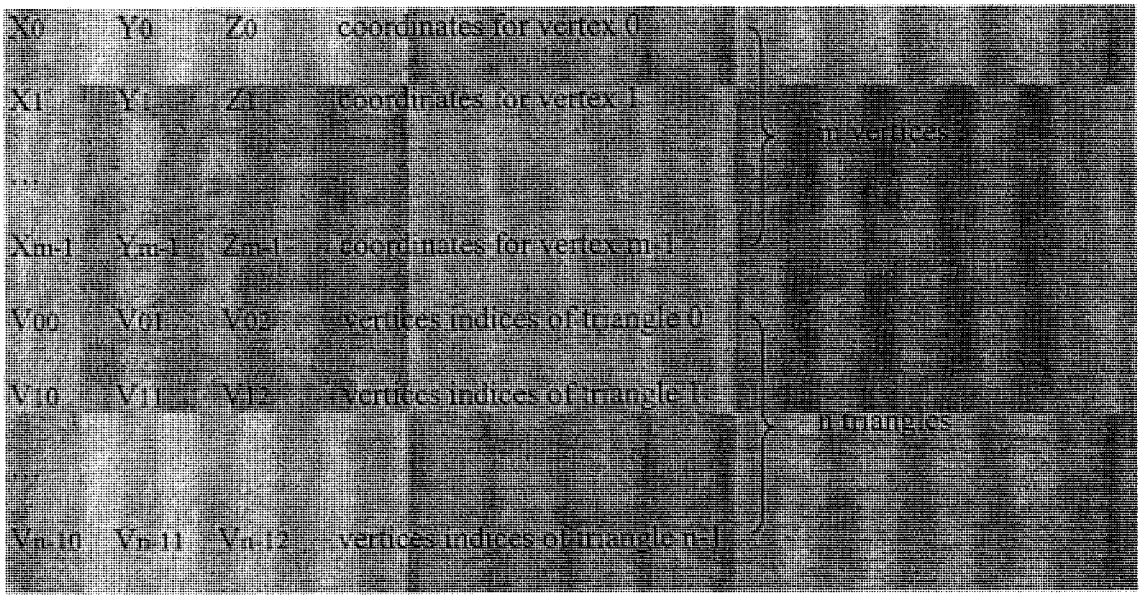
Figure 24 shows the image packet structure. Image data are represented as RGB values. Camera ID, image ID and packet offset are three headers of the packet, where camera ID and image ID are used for the purpose of synchronization. Since the data of a single image is too big to be transmitted in one packet over network, it is divided into small RTP packets. Packet offset means the packet number of an image. Remote site uses this information to regenerate a whole image.

```
        H := 240+(R - G) ×60/S
if (H<0)
    H := H+360
// 0≤V≤1, 0<S≤1, 0≤H≤360. The values are then converted to the 8bit values
V := V×255
S := S×255
H := H/2

// Step 2: comparison between foreground hue values and background hue values
for each pixel
    if (|H - Hb|≤T)   // H is the foreground hue value, Hb is the pre-screen's hue value,
                      // about 60, T is the threshold
        Y=0
    else
        Y=255

// Step 3: filtering noise (noise is some black pixels in the white area and some white
// pixels in the black area).
for each 3×3 pixels block
    if (the number of 255s< the number of 0s)
        Y=0
    else
        Y=255
```

4.5 3D Reconstruction

As shown in Figure 25, there are two threads at the reconstruction site. The main process receives silhouette packets. The reconstruction thread performs the 3D reconstruction and sends mesh data. The same method as the capture site is used to avoid conflicts between reading and writing. After receiving a frame from all cameras, the main process determines if the thread is reconstructing. If yes, the current foreground images are discarded.

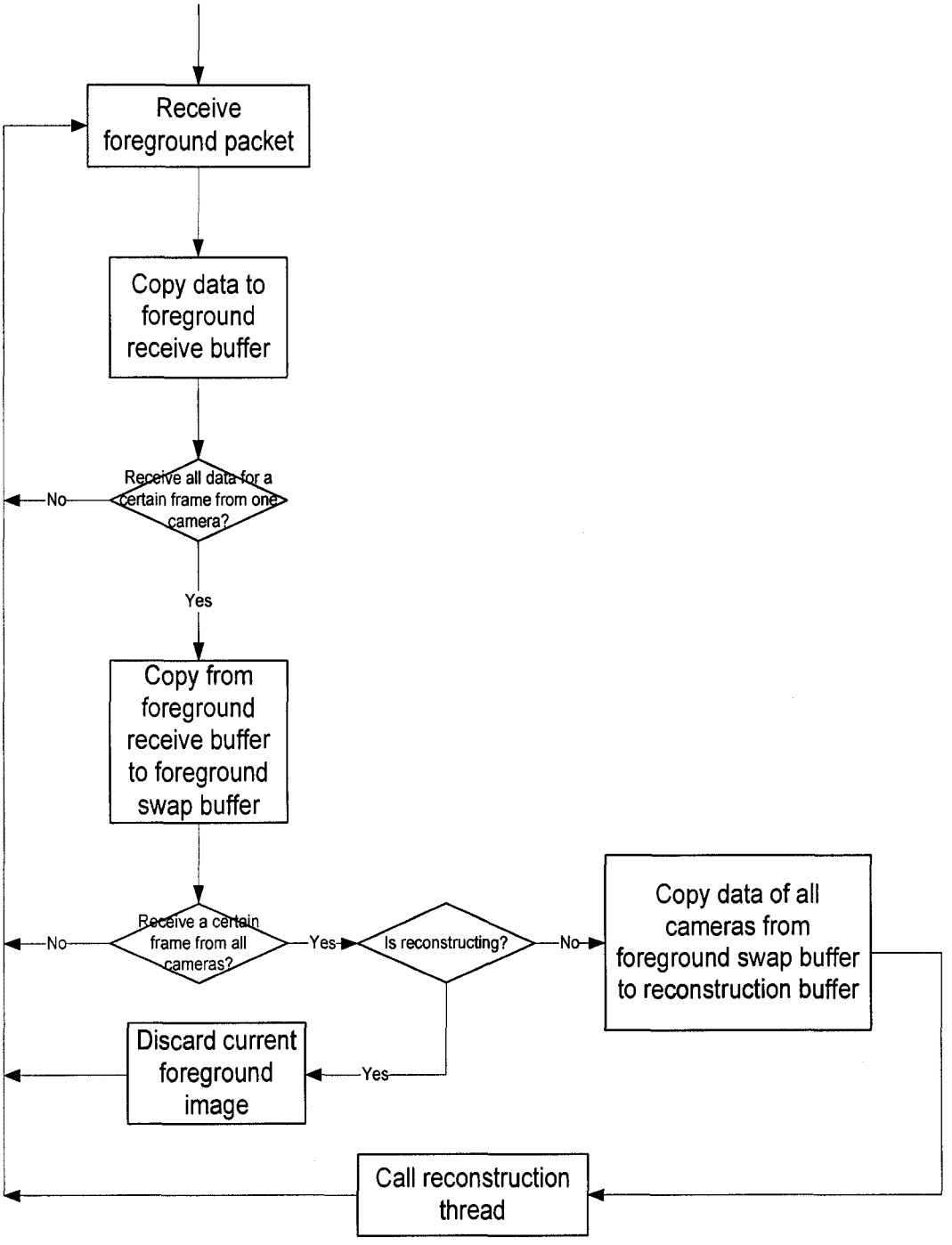
To reconstruct 3D mesh models, I use the Exact Polyhedral Visual Hull Library [50] which is built on Linux. The silhouettes have to be converted into polygons first. Then the four silhouette cones are intersected to get the polyhedral visual hull (detail algorithm in 3.2.2). The reconstruction outputs the 3D mesh data whose structure is as follows:



The mesh data are so big that they can not be transmitted as a whole packet. The mesh data are wrapped into three types of packets: information packet, point packet and triangle packet, which are shown Figure 26. The information packet includes the model ID, the number of points, the number of triangles and the number of edge (not used). The point packet includes the point ID offset and points' coordinates. The triangle packet includes the triangle ID offset and vertices indices of triangles. The point ID offset encodes the starting point ID of the point packet. Triangle ID offset encodes the starting triangle ID of the triangle packet. The offsets are used to regenerate the mesh data at the remote rendering site.

4.6 Rendering and Display

Figure 27 shows the setup of the rendering and 3D display system at the remote site. For the users to be able to extend their arms and move their head in the projected space (for natural manipulation and visualization), three orthogonal rear-projection screens are used: a table at waist height, and two walls to form a corner (as shown in Figure 19). The screen is big enough (1.2m×0.9m) to show life-size stereo video. Each screen is illuminated by two LCD projectors (with 1024×768 resolution) through two circular polarizing filters. Each two projectors are driven by a workstation (3.06GHz dual-Xeon CPU with nVIDIA Quadro FX 3000 video card). Three workstations compose a rendering cluster controlled by a set of input and output device. The user wears polarized glasses to see the 3D immersive scene. The viewpoint is required for the view-dependent rendering. The system utilizes a conventional Ascension Flock of Birds six degrees-of-



Main process

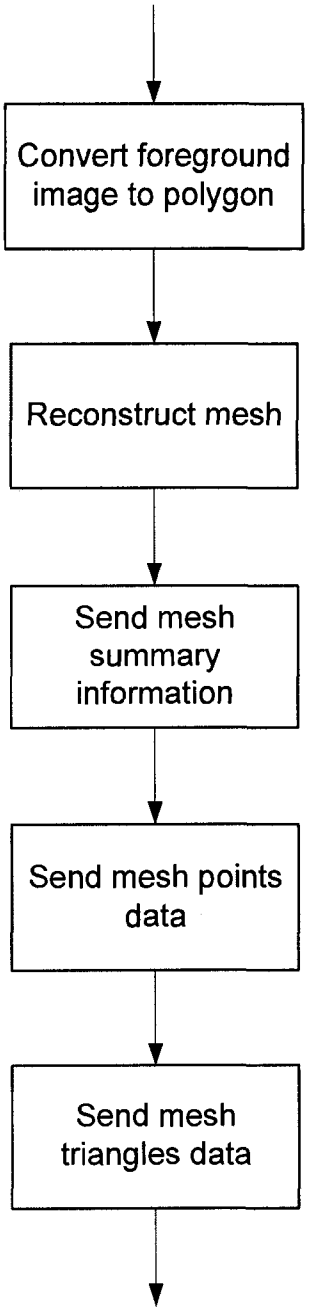


Figure 25: Flow chart of reconstruction site

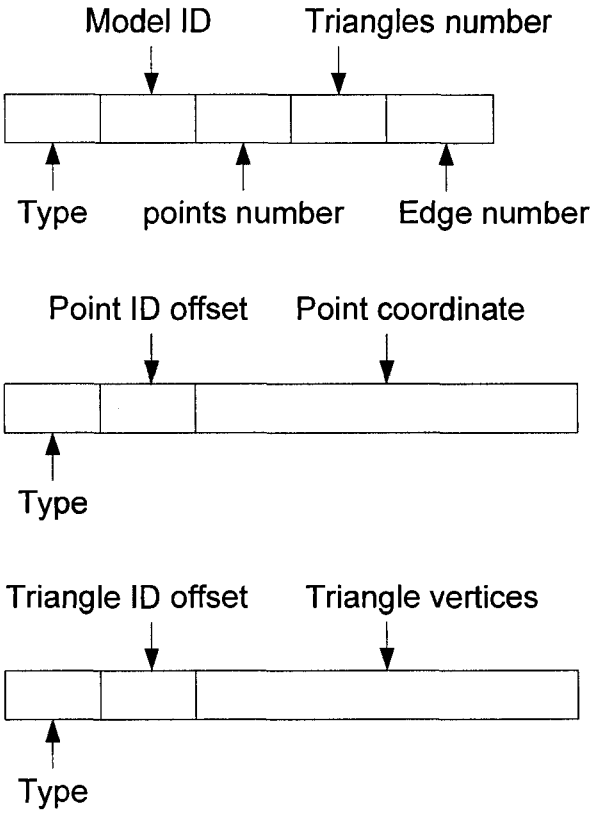


Figure 26: Mesh packets structures

freedom magnetic tracking system. The tracker gets user's head position from a magnetic sensor and sends the position information to one workstation through a RS232 cable. The head position information is shared with other two workstations over network. The mesh and texture data are received from the reconstruction site and the capture site respectively over gigabit Ethernet. The three workstations run the same application, so the graphics information is multicasted to the three machines.

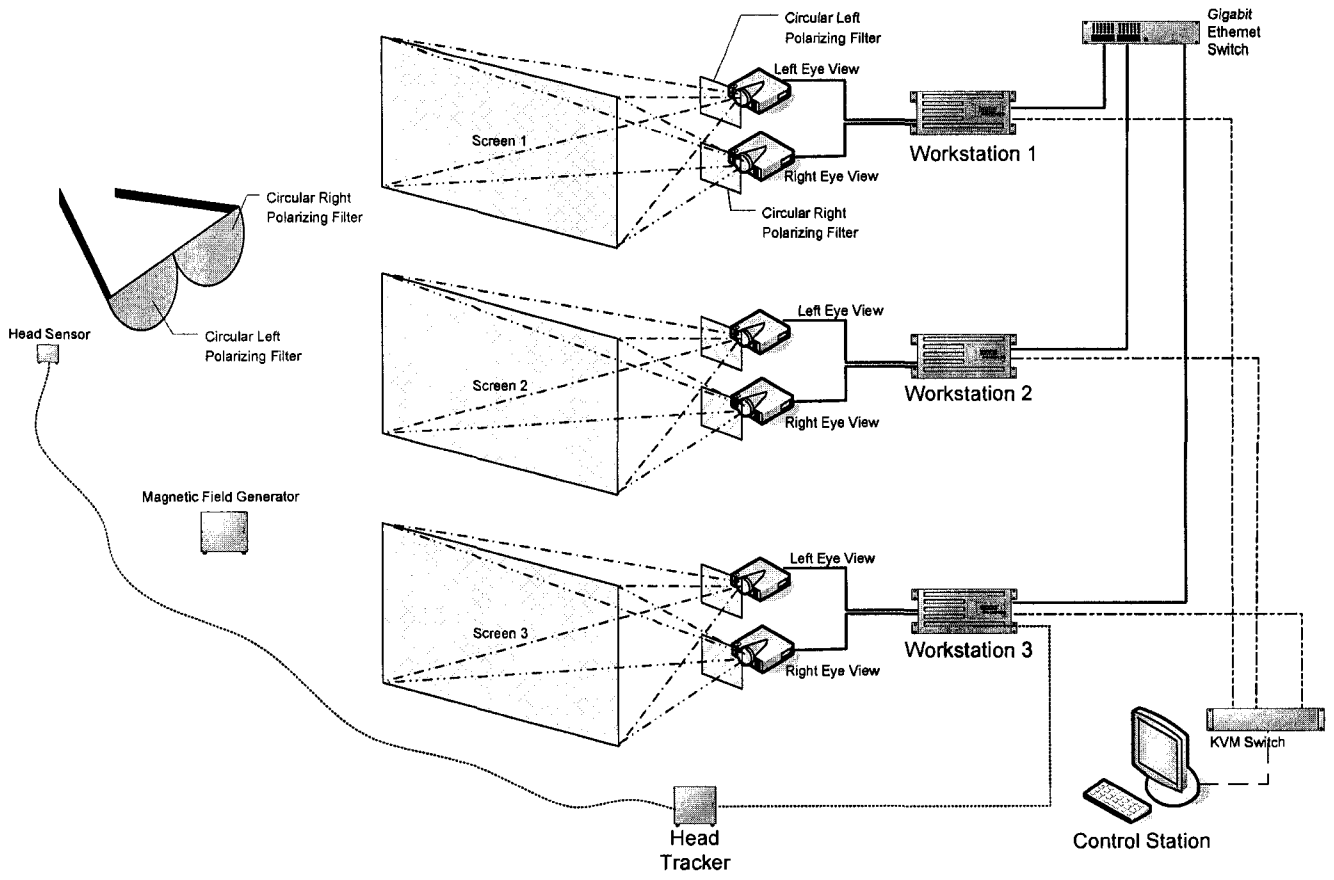


Figure 27: Rendering and 3D display system

The CAVElib API [53] and OpenGL are used to draw graphics and perform the view-dependent texture mapping. CAVElib can take care of the low level rendering work such as off-axis perspective transformation for the display device, synchronization between multiple display processes, drawing stereoscopic views and sharing data between cluster nodes.

Each workstation's application has three threads. One receives meshes (see Figure 28), one receives texture (see Figure 29) and one performs the view-dependent texturing and display (see Figure 30). For the purpose of the synchronization between mesh and texture

in the rendering thread, the model ID and the texture ID are compared. If the model ID is equal to the texture ID, the rendering thread draws the new model; else, it draws the old model.

The coordinates of the view point are acquired from the sensor on the user's head at each frame before drawing. Texture mapping is performed based on that view point. So when the user moves his or her head, the view changes. The detailed algorithm is in 3.3.2. A technical detail about texture mapping that should be mentioned is that in OpenGL, the height and width of the texture image must be equal and a power of 2. The 640×480 texture images captured from cameras have to be enlarged to 1024×1024 pixels by adding pixels to the right and top (shown in Figure 31).

At any place where read and write conflicts may occur, I put a MUTEX. MUTEX (**MUT**ually **EX**clusive Lock) is a programming flag used to avoid the simultaneous use of a common resource. The MUTEX is set to "lock", which blocks other attempts to use the same resource. The MUTEX is set to "unlock" when the data is no longer used. For example in the mesh receiving thread, before copying date from mesh receive buffer to mesh swap buffer, the write lock for mesh swap buffer is set. In the rendering thread, before copying data from mesh swap buffer to mesh draw buffer, the read lock for mesh swap buffer is set.

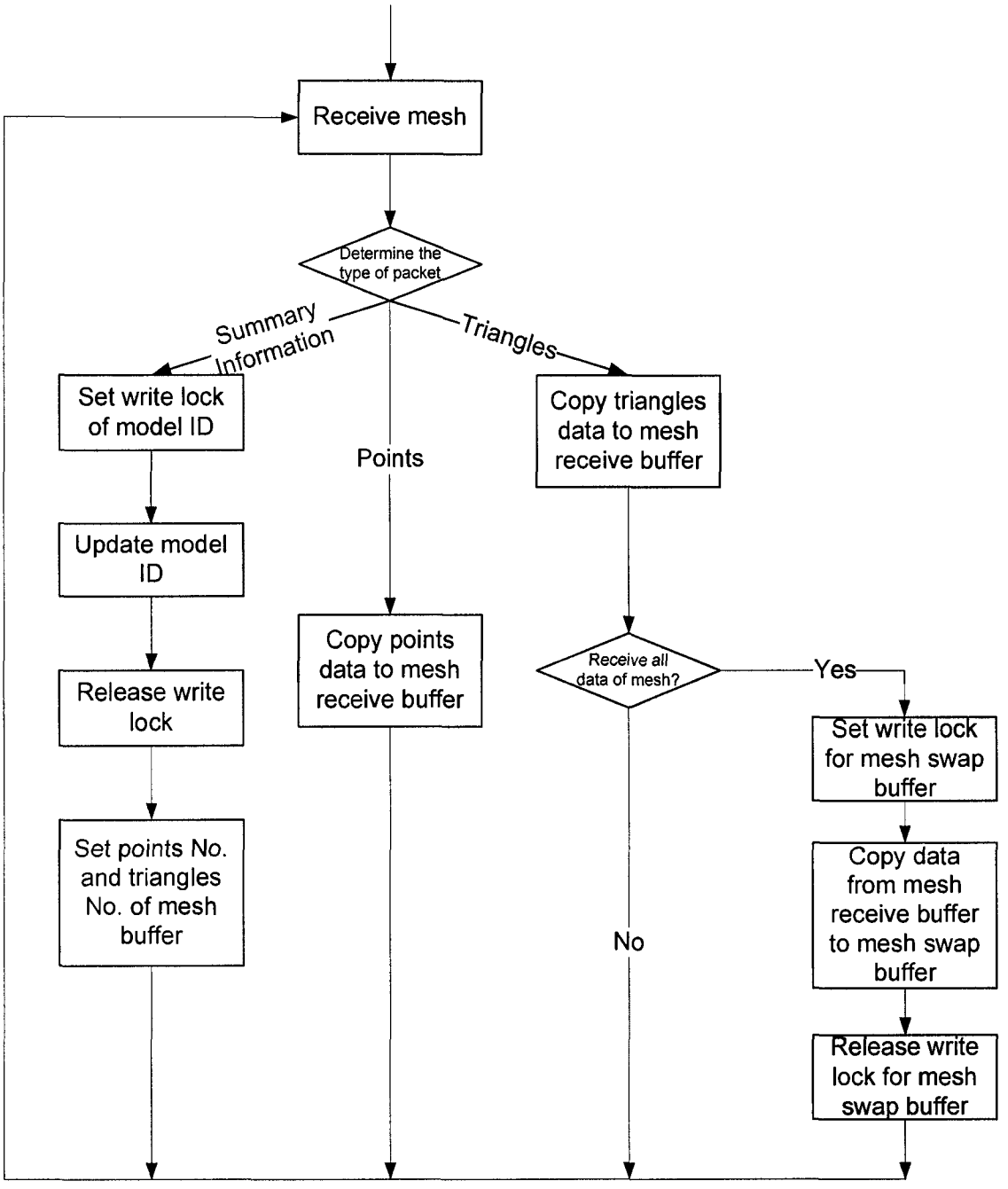


Figure 28: Flow chart of mesh receiving thread

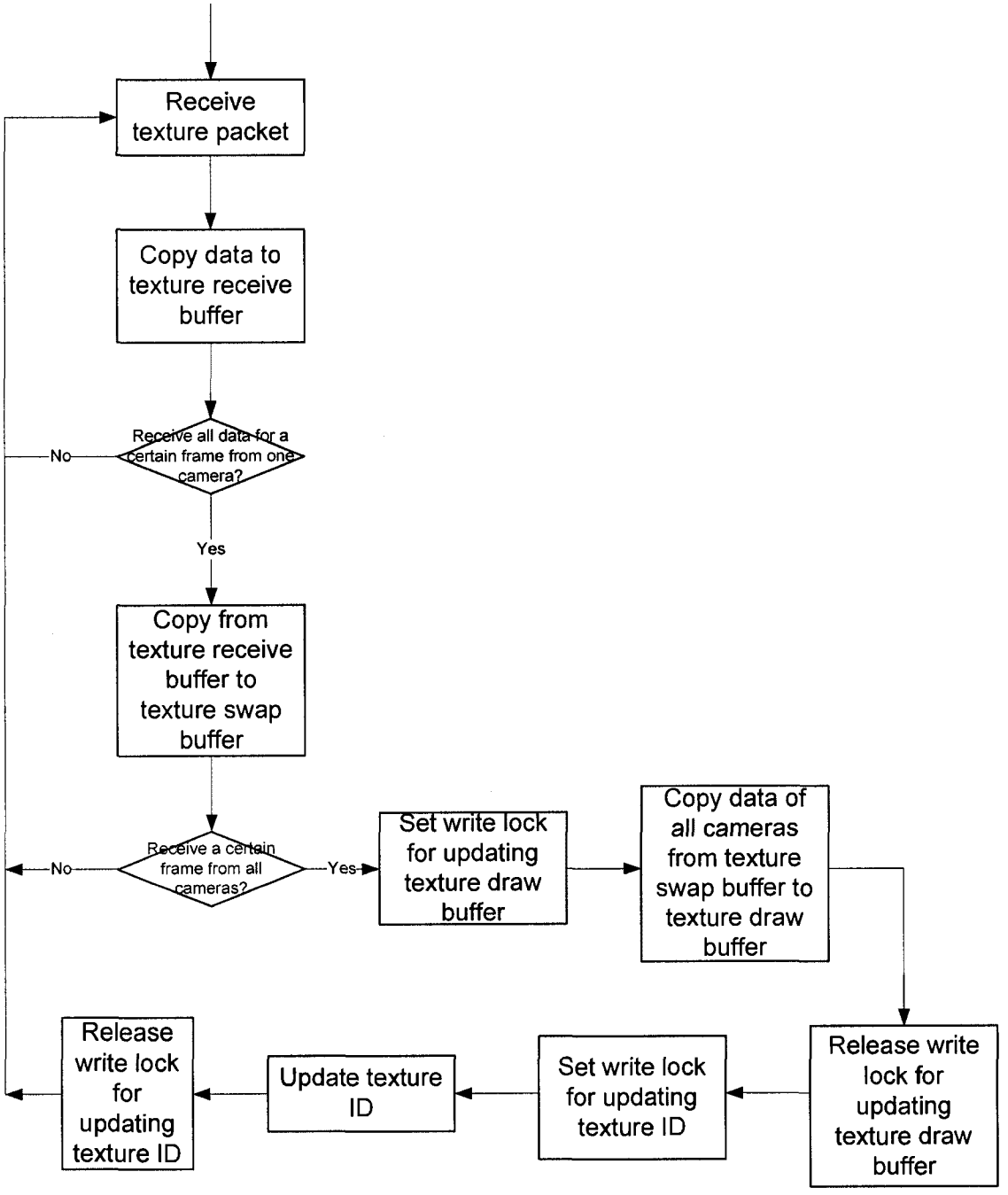


Figure 29: Flow chart of texture receiving thread

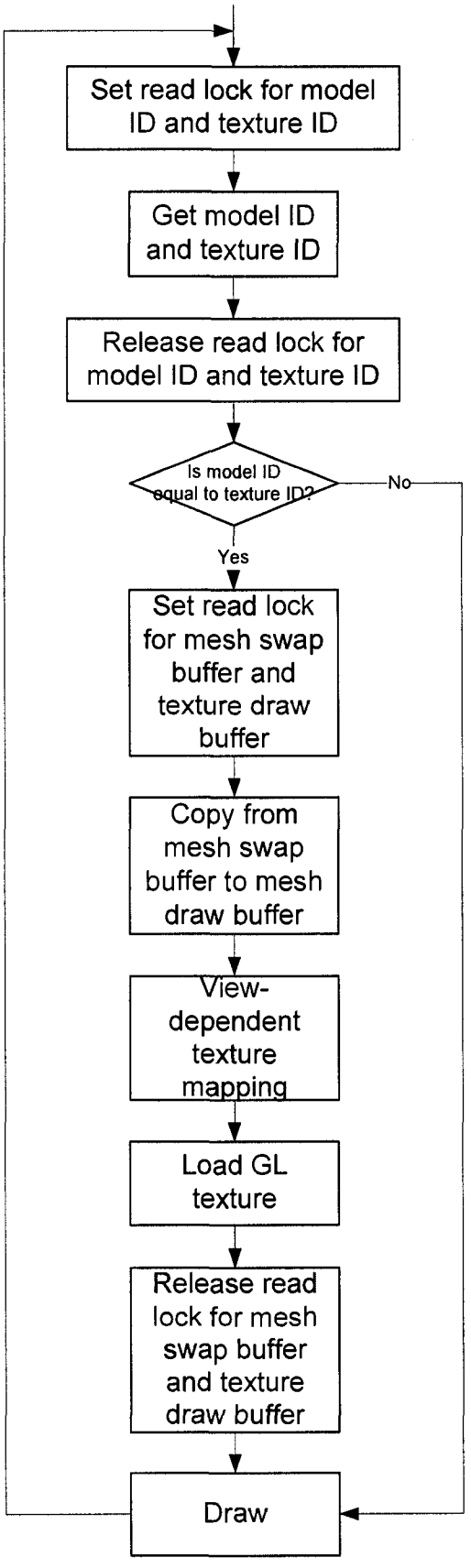


Figure 30: Flow chart of rendering thread

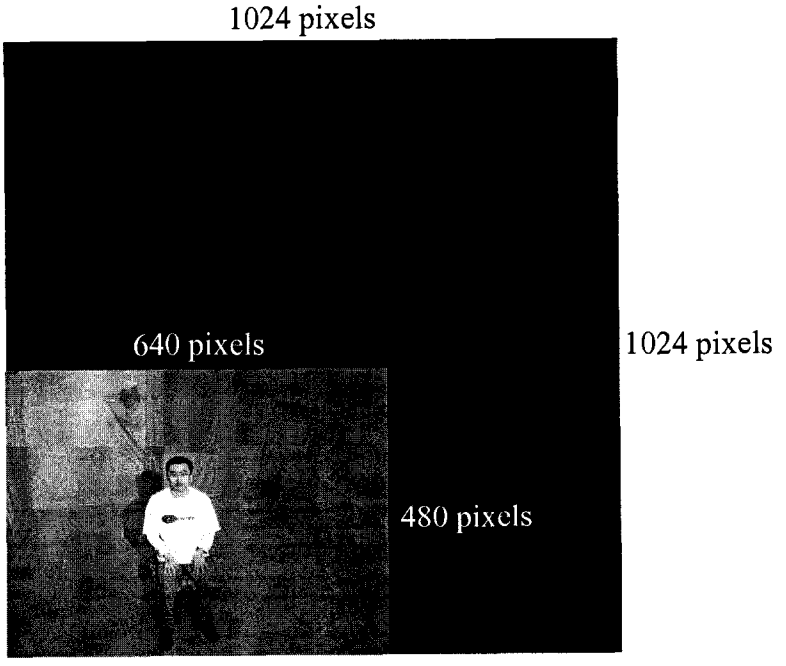


Figure 31: Enlarging texture image from 640×480 to 1024×1024

Chapter 5

Results and Analysis

In this chapter, the overall system performance is tested over a Gigabit LAN at the DISCOVER lab. The visual result is evaluated from a subjective perspective and analyzed theoretically. A major performance parameter, frame updating rate at the rendering site is analyzed. Through the tests, the reason of low frame updating rate is found and the solutions are then given.

5.1 Visual Result

The visual result shown in Figure 32 demonstrates that our system provides 3D tele-immersion experience. With passive stereo glasses, life-size 3D images of the remote conferee "float" over the three stereo displays, as if he was sitting there. The view is changed according to the position of the head-mounted tracker.

Figure 32: Visual Result

The quality of the models and texture are critical to the visual quality of stereo video. In the experiments, the model on the stereo displays looks "stronger" than the real person due to the lack of a precise geometry. The visual hull algorithm only reconstructs a maximal volume which is guaranteed to contain the object. The texture looks not very clear because of projective texture mapping and low resolution of texture images. To show the stereo images on the 1.2m×0.9m screen (with 1024×768 resolution) in life-size, the texture images with 640×480 resolution are magnified to fit for the life-size model. In other words, to show the model rendered from the virtual camera on the 1.2m×0.9m screen in life-size, the distance from the virtual camera to the model is shorter than the

distance between the real cameras and the model. This is equal to magnifying the texture images.

As shown in Figure 33-39, with four cameras setting, the quality of the reconstructed 3D shapes is good enough for the video conferencing application. But the geometry is not accurate for 3D modeling. More cameras are required to further refine the geometry. In addition, texture can be improved by using cameras with high resolution. Both operations will make the computation load costly.

Figure 33: Four captured images of the conferee

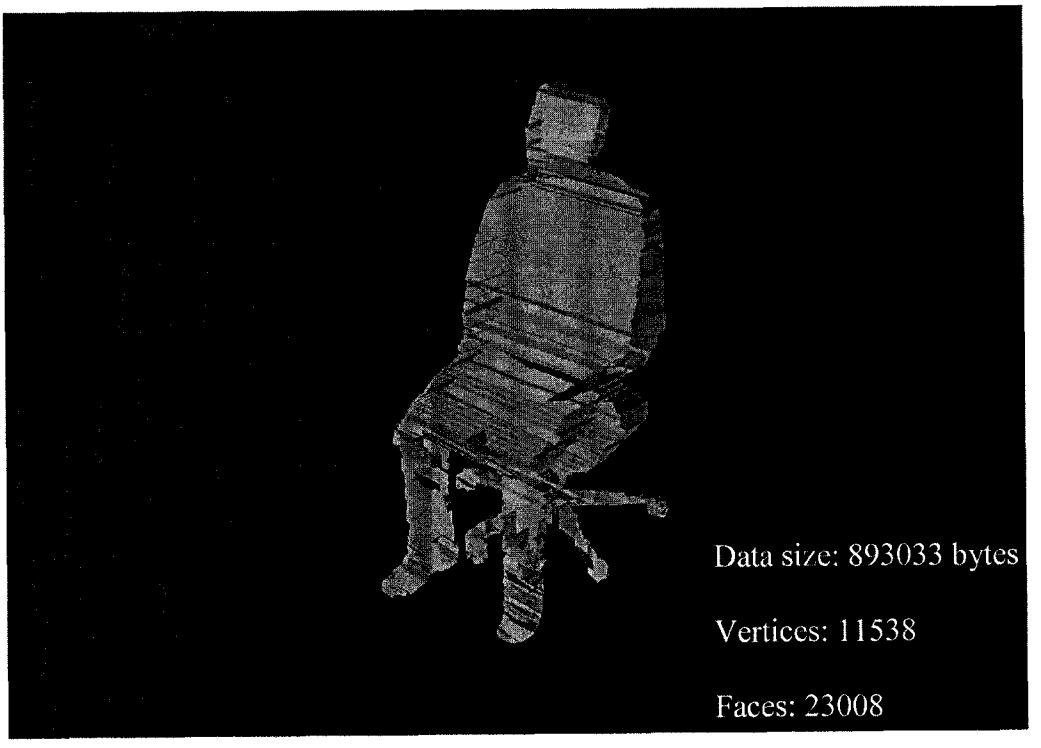


Figure 34: 3D model of the conferee

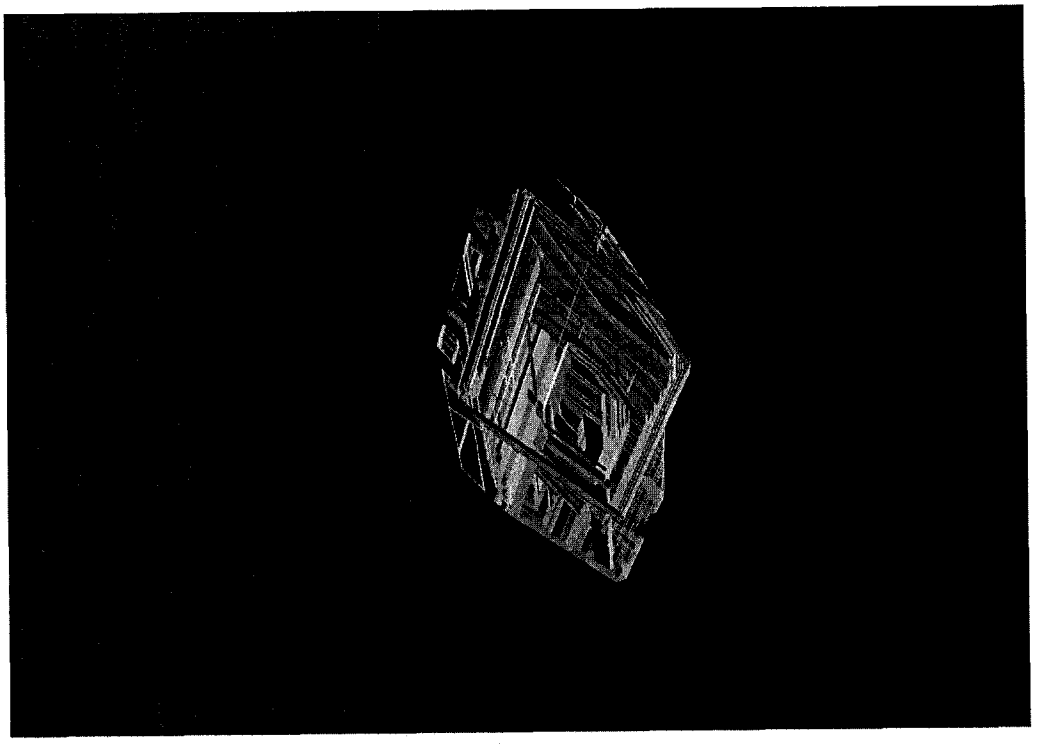


Figure 35: Planform of the 3D conferee

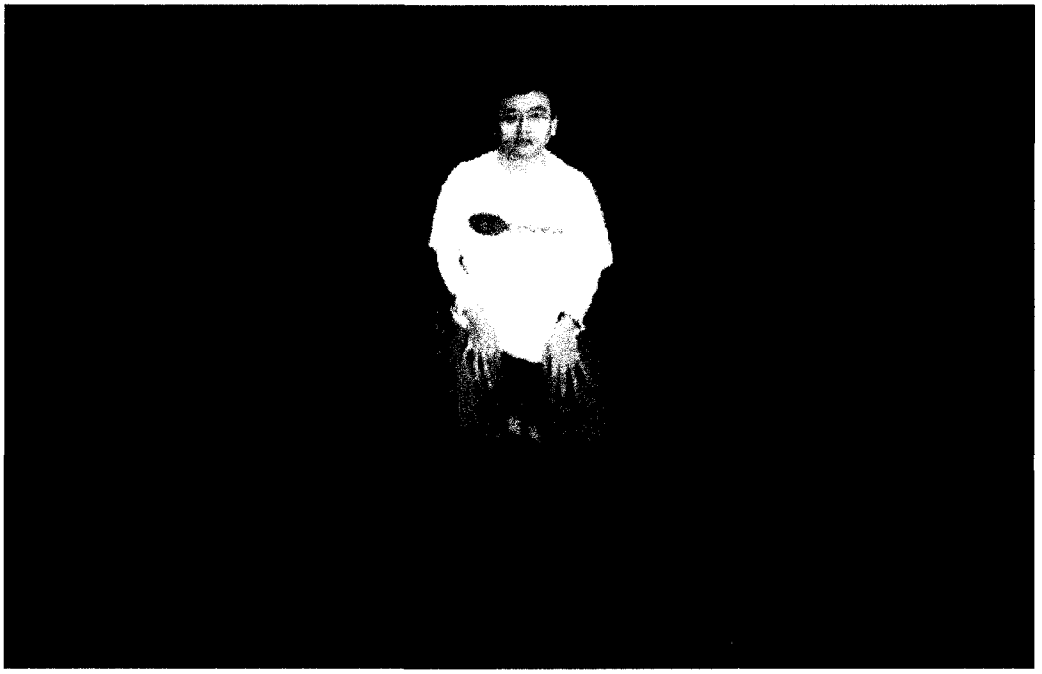


Figure 36: 3D model of the conferee with texture

Figure 37: Four captured images of a pot

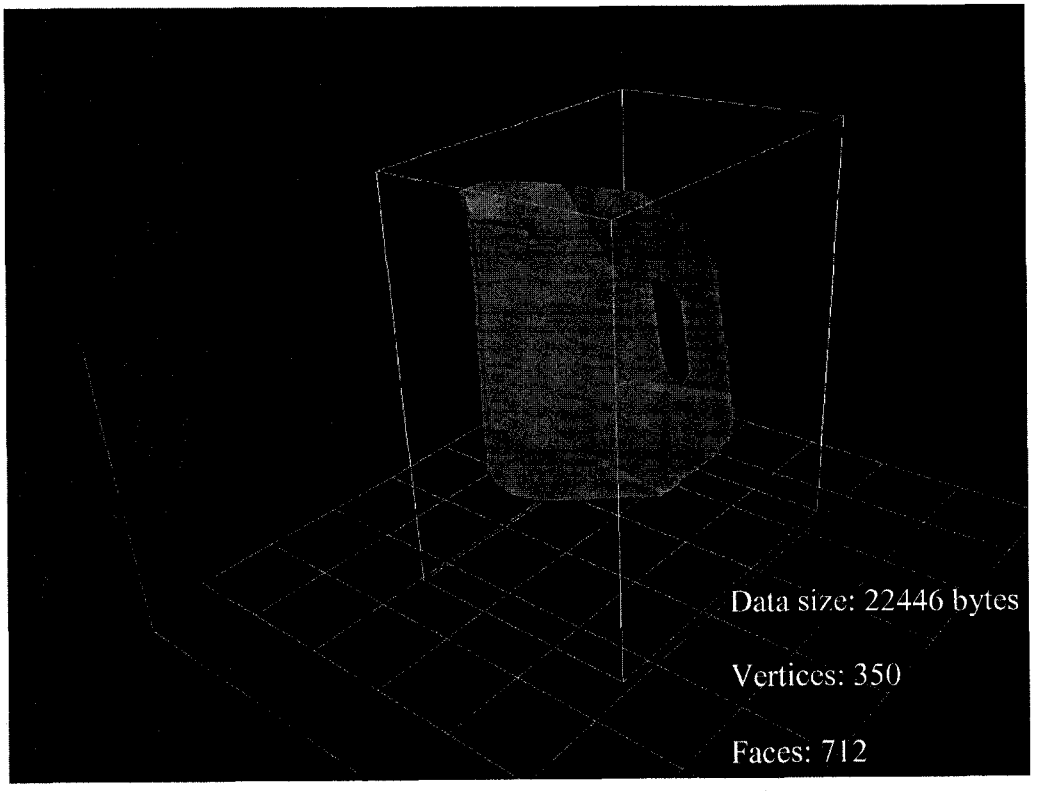


Figure 38: 3D model of the pot

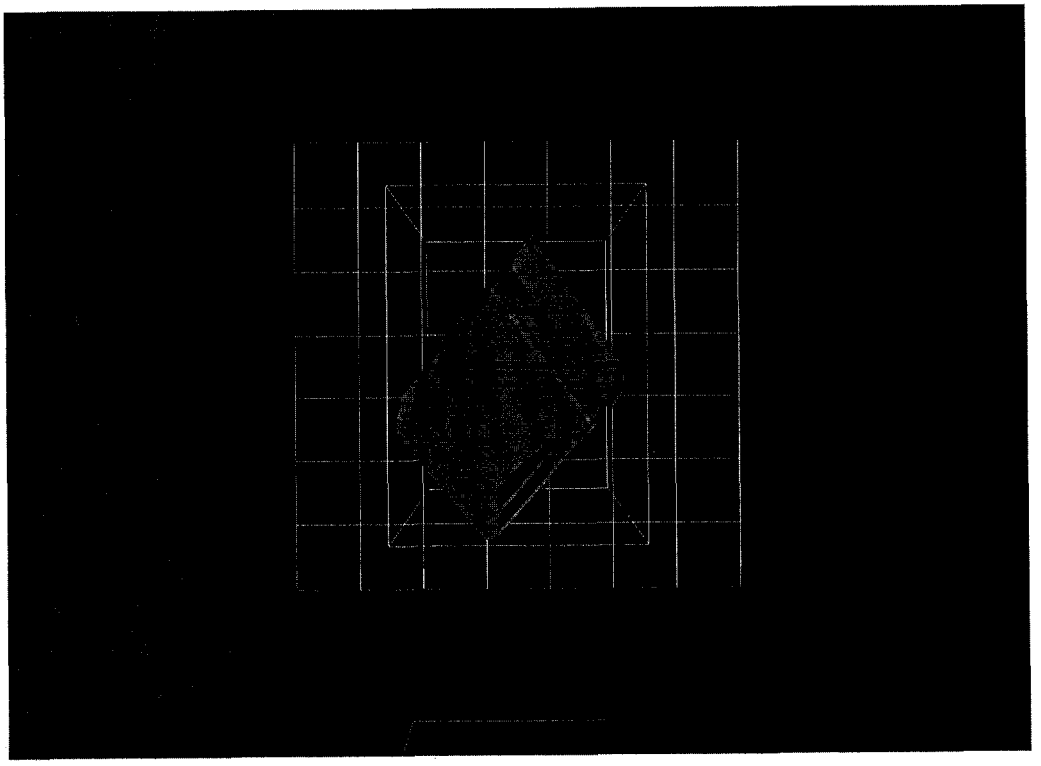


Figure 39: Planform of the 3D pot

5.2 Performance

At the rendering site, the models update at 3 frames per second, which is much slower than the camera's capture rate 15 f/s. 3D reconstruction speed is 40~100 ms per frame (depending on the model complexity), which means that the reconstruction thread can process at least 10 frames per second. The problem comes from the capture site: 3 frames are sent to the network per second. In other words, there are 4 cameras and 3 channels of color texture image data and 1 channel of grey silhouette image data are sent for each camera, so the network utilization is only $\frac{4\times 3\text{f/s}\times 640\times 480\text{pix/f}\times 4\times 8\text{b/pix}}{1\text{Gb}}=12\%$, which is also shown in the networking monitor of Windows Task Manager. Why is the network utilization so low?

5.3 Test 1

To determine if the problem comes from capture or comes from RTP, I made test 1 in which a dead loop is created. At each loop one RTP packet is sent. The RTP packet size can be adjusted. The data to be sent is the fixed data which has already been in the memory. The data does not come from cameras, so it does not take time to generate data. Test 1 tests the performance of Gigabit Ethernet with packets of different sizes using RTP or UDP.

First, the gigabit Ethernet is tested using RTP on a computer which has two 3.06GHz processors. The result is shown on the blue curve in Figure 40. Small packets (1024Bytes)

can only get 18% network utilization. As the packet size increases, the network utilization climbs up and at 65000B gets a peak 86% which is the bottleneck of PCI-slot network cards. Then, the network is tested using UDP and a similar result is obtained, which is shown on the yellow curve. The result demonstrates that the fact that the network utilization is proportional to the packet size is not the characteristic of RTP, but the characteristic of the gigabit network. A typical application of a gigabit network is not on the desktop but as backbone connection between central servers. High utilization is realized by the fact that multiple users share a gigabit network. This scenario is also simulated by running two and three programs to send small RTP packets (1024B) through different sockets. Network utilization 23% and 30% are obtained respectively.

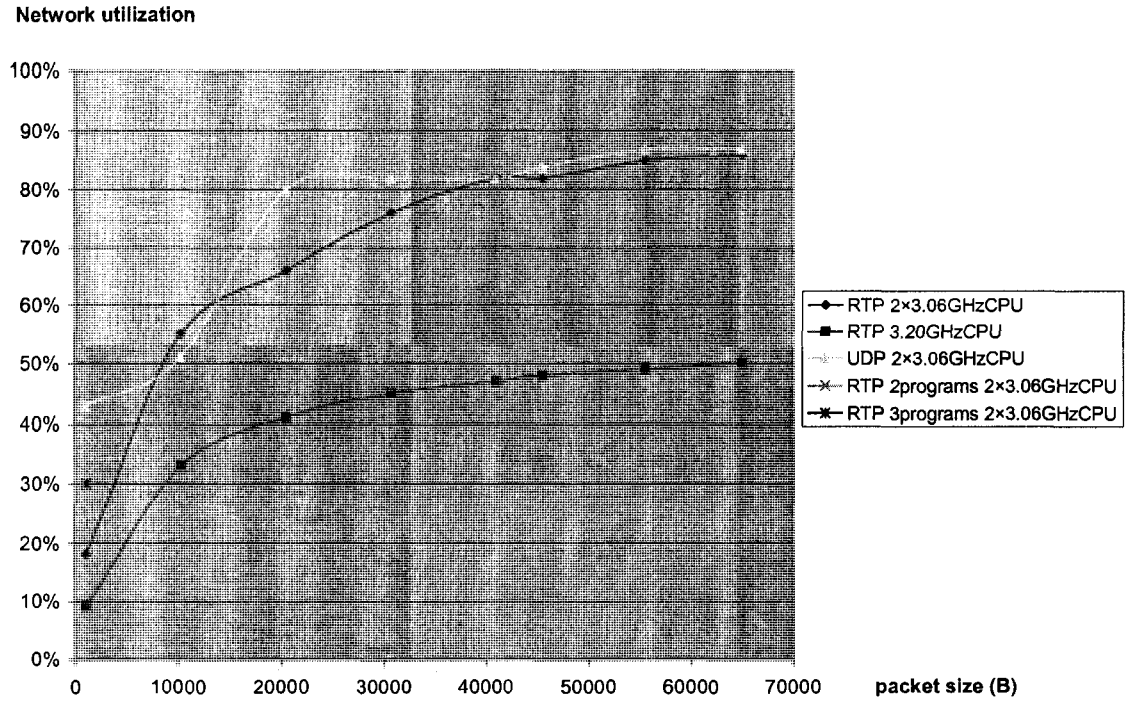


Figure 40: Test 1 Results: Gigabit Ethernet utilization based on an increasing packet size distribution

Why can only large packets get high network utilization? In one sending loop, RTP not only sends the packet, but also computes the timestamp and monitors the network. So it takes time to put the data to the network card. When sending small packets, there are lots of spaces left on the network, so the network utilization is low. To further prove this explanation, I use a slower machine (3.20GHz CPU) to test. As one RTP sending loop takes more time, the network utilization is lower, which is shown on the pink curve.

5.4 Test 2

From the test 1 the conclusion arrived at is that using large packets can realize high network utilization. Large packets (25600B instead of 12800B) are tested on the 3D teleconferencing system, which is test 2.

The network utilization increases from 12% to 15% which is still not high enough to send all captured image data. To find the reason for the low utilization, the sending program is tested using different image capture rates with or without the image segmentation process. The results are shown in Figure 41. Supposed network utilization is the theoretical utilization if all captured data is sent.

Frame capture rate (f/s)	Data capture rate (b/s)	Supposed network utilization	Practical network utilization with image segmentation	Practical network utilization without image segmentation
3.75	110,592,000	11%	12%	12%
7.5	221,184,000	22%	13%	15%
15	442,368,000	44%	15%	18%

Figure 41: Test 2 Results: network utilization with different frame capture rates

There are three conclusions drawn from the results. First, the image segmentation takes time, making one RTP sending loop longer. If we remove the image segmentation process, network utilization can increase from 15% to 18%. Second, it takes more time to send an image than to capture one. Capturing waits for sending. In the main process (see Figure 23), every time after capturing a group of four new images, the transfer thread is checked if it is transferring the last group of four images. If yes, the new group of images is discarded. As a lot of images are discarded, the sending rate is only 3f/s. Third, sending also waits for capturing. The worst case is that it takes 1.1 frame capture time to send a frame. After capturing the second group of images, the transfer thread is still sending the first group of images and the second group of images is discarded. After sending the first group of images, the transfer thread waits for the third group of images. Sending loop therefore idles for 0.9 frame capture time. This is the reason why increasing the frame capture rate can increase the network utilization a little bit.

5.5 Solutions

The reason of the low frame updating rate (3 f/s) at the rendering site is that 3 frames are sent to the network per second at the capture site. To increase the speed of sending packets, the algorithm of sending packets needs to be improved, making the time of sending a frame shorter than the time of capturing a frame. To make the capture thread and the transfer thread run independently and not wait for each other, buffers are needed for captured images.

Ideally, the sending rate will be equal to the capturing rate, 15 f/s. At that time frame updating rate at the rendering site will be 10 f/s which will be based on the reconstruction speed.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In this thesis, the first steps towards an immersive 3D video conferencing system were implemented, which include dynamically reconstructing 3D mesh based models, transmitting them to the remote site through a gigabit Ethernet, rendering and showing them on the stereoscopic displays in real-time. The system is a novel distributed platform for real-time teleimmersion, which integrates many state-of-the-art technologies, such as video acquisition, image segmentation, 3D reconstruction, texture mapping, and 3D display. The Exact Polyhedral Visual Hull algorithm can robustly fulfill the efficient 3D reconstruction, which is required for the real-time 3D video communication. View-dependent rendering technique is employed, allowing the user to perceive a free-viewpoint video. Half-cubic stereo display system called DIVINE provides users the sense of immersion. Performance analysis demonstrates that the implementation is a good framework toward the ultimate goal of 3D collaborative telepresence.

The major advantage of a 3D immersive teleconferencing is the natural interaction with the remote conferees. With the current display setup, users cannot easily establish eye-contact wearing the passive stereo glasses. Auto-stereoscopic displays are needed to make the current system more practical. With a tracker helmet on the user's head, the user encounters interference with the tracking system. An optical system or vision-based tracking will be utilized in the future.

Overall the application is prosperous and with continuous testing and modification it will eventually reach its goal of providing the users the fully immersive sense in a teleconference environment.

6.2 Future Work

There are many different avenues for the future work. The next step of the research would be the implementation of full-duplex communication mode, as the current prototype works in a half-duplex mode. Another consideration is that in order not to limit the application to a specific environment (green screen as the background), background subtraction techniques could be used to segment the images. In addition, image compression technologies, such as JPEG and MPEG4, could be employed to make the application feasible for a 100Mb/s network. Furthermore, audio acquisition and reproduction could be realized to make 3D video conferencing natural and comfortable. What is more, a computer graphic virtual meeting environment could be built to provide a graphical user an interface to the virtual world for use during a session. Applications

like collaboration between conferees at different locations to manipulate shared objects in a virtual environment, virtual architecture guidance and collaborative games could be envisaged.

References

[1] P. Eisert, "Immersive 3-D Video Conferencing: Challenges, Concepts, and Implementations," *Proc. SPIE Visual Communications and Image Processing VCIP-2003*, Lugano, Switzerland, pp. 69-79, July 2003

[2] AG Alliance: "Access Grid", home Page, http://www-fp.mcs.anl.gov/fl/accessgrid/

[3] VRVS: "Virtual Room Video-Conferencing System", Home Page, http://www.vrvs.org/About/index.html

[4] M. Chen, "Design of a virtual auditorium," *Proc. of ACM Multimedia*, (Ottawa, Canada), Sep. 2001.

[5] Fuqua School of Business: "Global Conference System", Press Release, Duke University, May 2002, http://www.fuqua.duke.edu/admin/extaff/news/global_conf_2002.htm

[6] D+S Sound Labs Inc.: "The Plasma-Lift A/V Conference Table", http://www.dssoundlabs.com/

[7] Home Page of Teleportec Ltd.: www.teleportece.com

[8] O. Schreer and P. Kauff, "An immersive 3D video-conferencing system using shared virtual team user environments," in *ACM Collaborative Environments, CVE 2002*, Bonn, Germany, 2002

[9] K. Okada, F. Maeda, Y. Ickikawaa and Y. Matsushita, "Multi-party videoconferencing at virtual social distance: MAJIC design," *Proceedings of CSCW'94*, 385-393, 1994

[10] T. Aoki, W. Kustarto, N. Sakamoto, N. Suzuki, K. Saburi, and H. Yasuda, "MONJUnoCHIE system: Videoconference system with eye contact for decision making," *Proc. International Workshop on Advanced Image Technology (IWAIT)*, 1999.

[11] S. J. Gibbs, C. Arapis, and C. Breiteneder, "TELEPORT - Towards immersive copresence," *Multimedia Systems*, 1999.

[12] W. Chen, H. Towles, L. Nyland, G. Welch, and H. Fuchs, "Toward a compelling sensation of telepresence: Demonstrating a portal to a distant (static) office," in *Proc. Visualization 2000*, pp. 327–333, (Salt Lake City, USA), Oct. 2000.

[13] H. Towles, W.-C. Chen, R. Yang, S.-U. Kum, H. Fuchs, N. Kelshikar, J. Mulligan, K. Daniilidis, L. Holden, B. Zeleznik, A. Sadagic, J. Lanier: "3D Tele-Collaboration Over Internet 2," Int. Workshop on Immersive Telepresence (ITP2002), Juan Les Pins, France, 6th December 2002

[14] O. Schreer, E. Hendriks, J. Schraagen, "Virtual team user environment (VIRTUE) – a key application in telecommunication," *Proceedings of eBusiness and eWork*, Prague, Cech Republic, CD-ROM, 2002

[15] R. Tanger, P. Kauff and O. Schreer, "Immersive meeting point (im.point) – an approach towards, immersive media portals," *Proceedings of Pacific-Rim Conference on Multimedia*, Tokyo, Japan, Part 1, 89-96, 2004

[16] TeleSuite 2004. Virtually anywhere. www.telesuite.com

[17] H. H. Baker, N. Bhatti, D. Tanguay, I. Sobel, D. Gelb, M. E. Goss, J. MacCormick, K. Yuasa, W. B. Culbertson, and T. Malzbender, "Computation and performance issues in coliseum: an immersive videoconferencing system," in

MULTIMEDIA '03: Proceedings of the eleventh ACM international conference on Multimedia, pp. 470–479, ACM Press, (New York, NY, USA), 2003.

[18] D. McLeod, U. Neumann, C. L. Nikias and A. A. Sawchuk, "Integrated media system: the move towards media immersion," *IEEE signal Processing Magazine* 16(1), 33-76. 1999

[19] M. Gross, S. Würmlin, M. Naef, E. Lamboray, C. Spagno, A. Kunz, E. Koller-Meier, T. Svoboda, L. Van Gool, S. Lang, K. Strehlke, A. Vande Moere, O. Staadt, "blue-c: A Spatially Immersive Display and 3D Video Portal for Telepresence," *Proceedings of ACM SIGGRAPH 2003*, pp. 819-827, July 2003

[20] W. Matusik, H. Pfister, , "3D TV: A Scalable System for Real-Time Acquistion, Transmission and Autostereoscopic Display of Dynamic Scenes," *ACM Transactions on Graphics (TOG) SIGGRAPH*, ISSN: 0730-0301, Vol. 23, Issue 3, pp. 814-824, August 2004

[21] Z. Yang, K. Nahrstedt, Y. Cui, B. Yu, J. Liang, S.Jung, R. Bajscy, "TEEVE: The Next Generation Architecture for Tele-immersive Environment," *ism*, pp. 112-119, Seventh IEEE International Symposium on Multimedia (ISM'05), 2005

[22] S. M. Rhee, R. Ziegler, J. Park, M. Naef, M. Gross, M. H. Kim, "Low-Cost Telepresence for Collaborative Virtual Environments," *IEEE Transactions on Visualization and Computer Graphics*, Volume 13, Issue 1, pp. 156-166, January 2007

[23] http://homepages.inf.ed.ac.uk/rbf/CVonline/geom.htm

[24] M. Bichsel, "Segmenting Simply Connected Moving Objects in a Static Scene," *IEEE PAMI* 16, 11, 1138-1142, November 1994

[25] N. Friedman and S. Russel. "Image Segmentation in Video Sequences," *Proc 13th Conference on Uncertainty in Artifical Intelligence*, 1997

[26] T. Horprasert, D. Harwood, and L.S. Davis, "A Statistical Approach for Real-time Robust Background Subtraction and Shadow Detection," *IEEE ICCV'99 frame-rate workshop*, 1999

[27] T. Kanade and M. Okutomi, "A stereo matching algorithm with an adaptive window: Theory and experiment," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 16(9), 920-932, 1994

[28] H. Moravec, "Robot rover visual navigation," *Computer Science: Artificial Intelligence*, No. 3, pp. 105-108, 1980/81

[29] D. Marr and T. Poggio, "A theory of human stereo vision," *Proceedings of the Royal Society London* B204, 301-328, 1979

[30] L. Boroczky, "Pel-recursive Motion Estimation," Department of Electrical Engineering, Delft University of Technol, 1991

[31] J. Barron, D. J. Fleet and S. Beauchemin, "Performance of optical flow techniques," *International Journal of Computer Vision* 12(1), 43-77, 1994

[32] B. K. P. Horn and B. G. Schunck, "Determining optical flow," *Artificial Intelligence* 17, 185-204, 1981

[33] A. Laurentini, "The visual hull concept for silhouette-based image understanding," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1994

[34] M. Potmesil, "Generating Octree Models of 3D Objects from their Silhouettes in a Sequence of Images," *CVGIP,* 40, pp. 1-29, 1987

[35] R. Szeliski, "Rapid Octree Construction from Image Sequences," *CVGIP: Image Understanding,* 58, 1, pp. 23-32, July 1993

[36] W. Matusik, C. Buehler, and L. McMillan. "Polyhedral Visual Hulls for Real-Time Rendering," In *Eurographics Workshop on Rendering*, 2001

[37] W. Matusik, C. Buehler, R. Raskar, S. Gortler, L. McMillan, "Image-Based Visual Hulls," *SIGGRAPH 2000*, pp. 369-374, July 2000

[38] J. S. Franco, E. Boyer, "Exact Polyhedral Visual Hulls," *British Machine Vision Conference (BMVC'03)* Vol. I, p. 329-338, September 2003

[39] M. Segal, C. Korobkin, R. Van Widenfelt, J. Foran, and P. Haeberli, "Fast shadows and lighting effects using texture mapping," In *SIGGRAPH '92*, pp. 249–252, July 1992

[40] P. Debevec, C. J. Taylor and J. Malik, "Modeling and rendering architecture from photographs: a hybrid geometry- and image-based approach," *Computer Graphics (SIGGRAPH'96)*, pp. 11-20, August 1996

[41] P. Debevec, Y.-Z. Yu and G. Borshukov, "Efficient View-Dependent Image-Based Rendering with Projective Texture-Mapping," *9th Eurographics Rendering Workshop*, Vienna, Austria, June 1998

[42] C. Buehler, M. Bosse, S. Gortler, M. Cohen, L. McMillan, "Unstructured Lumigraph Rendering," *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, *SIGGRAPH '01*, 2001

[43] A.R.L. Travis, "The display of three-dimensional video images," *Proceedings of the IEEE*, Volume 85, Issue 11, Page(s):1817 – 1832, Nov. 1997

[44] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications," Internet Engineering Task Force, RFC 1889, January 1996

[45] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications," Internet Engineering Task Force, Work in Progress (Update to RFC 1889), March 2003

[46] H. Schulzrinne and S. Casner. "RTP Profile for Audio and Video Conferences with Minimal Control," Internet Engineering Task Force, Work in Progress (Update to RFC 1890), March 2003

[47] http://shape.cs.princeton.edu/benchmark/documentation/off_format.html

[48] http://www-out.bell-labs.com/project/RTPlib/

[49] http://research.graphicon.ru/calibration/gml-c-camera-calibration-toolbox-5.html

[50] http://www.vision.caltech.edu/bouguetj/calib_doc/

[51] http://www.intel.com/technology/computing/opencv/

[52] http://perception.inrialpes.fr/~Franco/EPVH/index.php

[53] http://www.vrco.com/CAVELib/HELP/index.html

[54] E. Trucco, and A. Verri, "Introductory techniques for 3-D computer vision," ISBN: 0-13-2611-8-2, Prentice-Hall, Inc. pp.143-144, 1998

[55] G. Medinoi and S. B. Kang, "Emerging Topics in Computer Vision," ISBN: 0131013661, Prentice Hall PTR, 2005

[56] M. Magnor, "Video-based Rendering," ISBN: 1568812442, A K Peters, 2005

[57] O. Schreer, P. Kauff, T. Sikora, "3D Videocommunication: Algorithms, concepts and real-time systems in human centred communication," ISBN: 047002271X, John Wiley & Sons, 2005

[58] http://www.tau.ac.il/~phchlab/experiments/Sucrose/Sucrose.htm

[59] http://blue-c.ethz.ch/index.php?menu=visuals&sub=pictures

[60] http://en.wikipedia.org/wiki/HSV_color_space

[61] PGR FlyCapture User Manual and API Reference, http://www.ptgrey.com/

[62] William Stallings, "Data and Computer Communications," ISBN-10: 0132433109, Prentice Hall, 2007

[63] http://www.zaxel.com/virtualviewpoint/