The sciences do not try to explain, they hardly even try to interpret, they mainly make models. By a model is meant a mathematical construct which, with the addition of certain verbal interpretations, describes observed phenomena. The justification of such a mathematical construct is solely and precisely that it is expected to work – that is, correctly to describe phenomena from a reasonably wide area.

– John von Neumann (1903-1957)

University of Alberta

Optimal Stereo Reconstruction and 3D Visualization

by

Hossein Azari

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Computing Science

Fall 2012
Edmonton, Alberta

Abstract

In this thesis different aspects of stereo-based 3D reconstruction and 3D visualization and manipulation are investigated. The focus of this research is specifically on sampled object representations (SOR) in which the object(s) is (are) shown using a collection of samples such as discrete pixels, voxels, points, or surface elements. We study different techniques of optimal sampling, and optimal representation and visualization of SORs for conventional and stereo-based 3D applications. In this regard, we first analyze the process of capturing and viewing stereo content on stereo capture and 3D display devices, and propose (unified) optimal sampling models for both the stereo capturing and viewing ends. Specifically, we theoretically show that for a given total resolution, a finer horizontal sampling rate, compared to the usual horizontally-vertically similar sample (pixel) distribution, results in a more accurate 3D estimation and enhanced 3D visual experience. We validate our theoretical results through subjective studies and show that human observers indeed have a better 3D viewing experience with an optimized *vs.* a non-optimized representation of stereo 3D content. We next study different techniques for compact representation and interactive rendering of sampled-based (point-based) 3D object representations. In particular, we introduce a tree structure, called a multi-section tree, and show how this structure can be used in creating a fully balanced, multi-resolution, hierarchical structure over space (and time) to support interactive rendering of spatial (or spatio-temporal samples) of still (or moving) 3D objects. Using an implicit representation of the multi-section tree and improved, dense hierarchical encoding techniques we

can achieve highly compact representations for both still and animated point-based 3D models. We can also achieve an interactive frame rate, and quality rendering of large models, on commodity desktop or mobile devices. Finally, we study 3D interaction and manipulation within a virtual stereoscopic 3D space. In this regard, based on a stereo-based 3D cursor, we have developed some simple 3D tools for manipulating point-based 3D objects. We discuss and through user studies show how this stereo-based cursor can be considered as a generic alternative to its 2D counterpart in virtual stereoscopic 3D space. We also discuss the pointing accuracy of the stereo-based 3D cursor with regard to the optimal sampling model proposed and discussed in the earlier chapters.

Acknowledgements

There are many people, organizations, and entities that I would like to thank for their assistance, services, and support. Without them many accomplishments in this thesis could be much more difficult to achieve, if not impossible.

Firstly, I would like to thank NSERC - Canada, AITF - Alberta, and the Department of Computing Science - University of Alberta, for their financial support throughout my PhD education.

Secondly, I wish to thank all those institutes and the people who have contributed in providing on-line, freely accessible 3D models, open source libraries, and data conversion and manipulation tools. Specifically, I would like to mention the Stanford Computer Graphics Laboratory, the CVU Center at Georgia Institute of Technology, the Computer Graphics Group of CSAIL-MIT, and Middlebury College for providing some of the 3D models, the 3D mesh sequences, and the stereo pairs that we have used in our algorithms, experiments, and user evaluations.

My special thanks and appreciation go to my supervisor Prof. Anup Basu for his kind support and guidance throughout this research. Many thanks to my co-supervisor Dr. Irene Cheng for her constructive advice and comprehensive comments during this research, and thanks to my supervisory and my PhD defense committee members for their time and their valuable comments on both my work and my thesis writing. Especially, I would like to thank Prof. Janusz Konrad for his thorough review and valuable feedback on my thesis writing. I also wish to thank MRC Lab members and all other people who dedicated their time and voluntarily participated in subjective user studies.

Lastly, I wish to express my deep gratitude and appreciation to my dear wife Parisa for her patience, understanding, and support during all these years. An honorable mention also goes to my family, especially my mother, for being totally encouraging and supportive, though they have been far far away, and also to my friends for making this journey more pleasant and memorable.

Table of Contents

List of Tables

List of Figures

List of Symbols

Notation	Definition
f	Focal length of camera
(X, Y, Z)	A 3D point
$(\hat{X}, \hat{Y}, \hat{Z})$	An estimated 3D point position
(x_l, y_l)	Projection of a 3D point on the left image plane/display screen
(x_r, y_r)	Projection of a 3D point on the right image plane/display screen
$e_x(e_y)$	Horizontal distance between two neighboring pixels (picture elements or viewing elements) in the $x(y)$ direction
b_x	Baseline of a stereo imaging/viewing setup
R	Total resolution, *i.e.*, the total number of pixels being used for stereo capture or viewing
d	3D display viewing distance
c, D, and h	Subscripts that are used to refer to a stereo imaging system (cameras), a stereo 3D display, and human eyes, respectively. For convenience, subscript c is often dropped from those equations that are referring to a standalone stereo capturing system.

Chapter 1

Introduction

The stereopsis capability of the human vision system is one of the primary means enabling us to have a 3D perception of the world. This ability provides us with precise information of the relative depth of the objects and a clear sense of their 3D shape from two slightly different images captured by the eyes [52]. Providing the same capability for machines, *i.e.*, the ability to reconstruct a united 3D model from two or multiple views of a scene, has been one of the most extensively researched topics in the field of computer vision [91, 93]. In this regard, a large number of algorithms have been proposed over the years and substantial progress has been made in different related areas or subproblems including techniques of matching corresponding regions or features, occlusion handling, multiview reconstruction, and real-time implementation [20]. Nevertheless, our 3D vision system in most cases is still far better than these algorithms in many aspects, including matching accuracy, flexibility, efficiency, and clarity of the resulting 3D models.

The human vision system is not only capable of reconstructing 3D models from 2D images directly captured by the eyes from a real 3D scene, but also often able to perfectly match stereo pairs that are captured or rendered using an external imaging device or rendering software. Benefiting from this ability, a wide variety of different stereoscopic devices have been applied to create the illusion of depth by presenting two separate, perspectively different views to the left and right eye. Traditionally most of these techniques required some sort of filtering glasses to separate the left and right channels, but recent advances enable users to watch stereo content in 3D without wearing any type of filtering glasses. These display devices, known as multiview, auto-stereoscopic 3D displays, present multiple views of the scene to the viewers, but only a pair of adjacent views is visible from each of the viewing zones surrounding the front side of the 3D display and only that pair contribute to

the formation of the 3D scene. By changing the viewing zone, viewers are able to perceive the scene from a different viewing angle, providing them with both stereo- and motion-parallax perceptual cues [37]. Despite these advances, there are still several challenges and/or deficiencies in both 3D content generation and visualization techniques and 3D device technologies that need to be addressed in order to have an ultimate, ubiquitous 3D experience [65].

Stereo reconstruction and stereo visualization are two sides of the same problem; hence, progress in one of them can be directly or subsequently beneficial to the other. The shared part of both problems is 3D modeling where the object is described in the form of a collection of constructive elements or, alternatively, in closed form representations. An appropriate 3D representation not only helps with efficient, quality rendering of the 3D object on the visualization side, but it can also be used as part of the pre-knowledge of a scene to enhance the 3D reconstruction results. Efficient, concise representation of a model can be significantly beneficial for efficient compression and transmission of the visual 3D data, as well.

In this thesis, broadly speaking, we study some of the aspects of stereo-based 3D reconstruction and 3D visualization, with a special focus on sampled object representations (SOR) in which the object(s) is/are shown using a collection of samples such as discrete pixels, voxels, points, or surface elements. These include studying methods of optimal sampling of stereo content for enhanced 3D reconstruction and visualization, and improved 3D modeling and representation techniques that provide us with "better" 3D visualized objects on conventional or (auto)stereoscopic display devices, while are more efficient in terms of the memory requirements and computational complexity. We also investigate methods of interaction with and manipulation of such 3D representations.

1.1 Motivation and Purpose

Over the past few years, stereo-based 3D display technologies have received increasing attention with wide applications in entertainment centers, home theaters, game consoles, and mobile devices. Research in this area covers a wide range of related topics including 3D capturing, 3D reconstruction and modeling, transmission, and display [73, 61, 92, 72, 13], as well as display design and technologies [56, 104]. Stereoscopic 3D display systems considerably elevate the spatial understanding of the 3D data by providing stereo-parallax and, in the case of multiview displays, motion-parallax perceptual cues for the human visual system. This en-

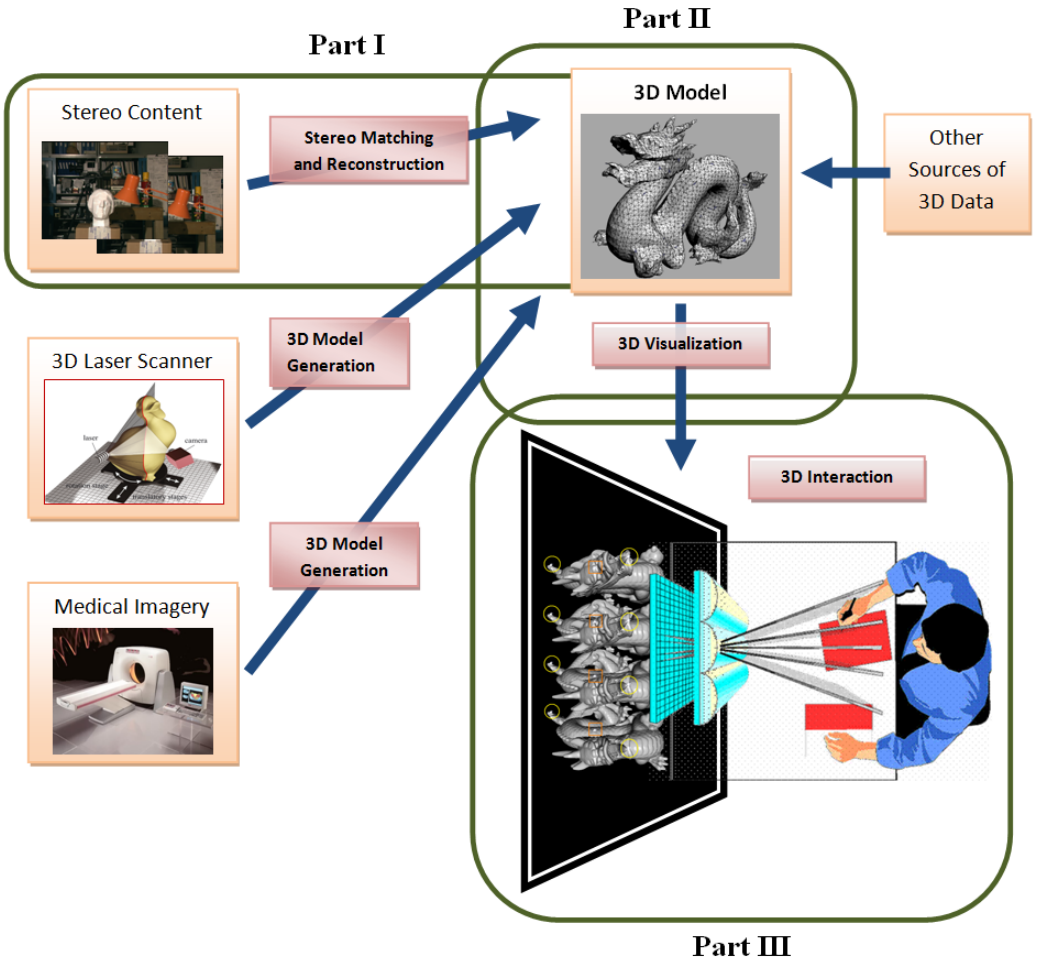


Figure 1.1: Typical components of a spatial or spatio-temporal 3D data generation and visualization system (parts of the figure are adopted from [16] and Wikipedia).

hanced representation of 3D data can be advantageous in providing a more attractive and also more comprehensible, interactive environment in many application areas including entertainment industry, medical data visualization, and educational applications. However, realizing these capabilities requires improved 3D data processing techniques and algorithms optimized for multiple, free view rendering and 3D interaction.

Stereoscopic displays may simply be used for watching raw stereo content such as stereo images or stereo video streams. Such raw stereo content can be fed into the display system with little processing effort, although compression and transmission may become a challenge for multiple channels (views), especially in the case of limited processing power and/or band-width. In this regard, the 3D picture formation process based on watching stereo images needs to be revisited and studied with

respect to the physical or psychophysical parameters affecting the whole process. This may include display design parameters such as the display screen aspect ratio and the relative horizontal *vs.* vertical display resolution (pixel aspect ratio), as well as different camera configurations or some other underlying depth perception parameters inherent in the human visual system that affect the quality of the 3D output delivered to the user. In fact, this is one of the main topics of this research which is investigated and discussed in Part I, where we try to mathematically formulate the stereopsis process and derive optimal sampling rates (optimal pixel aspect ratio) for a given band-width (image resolution). The results of these studies can be useful to 3D display design, as well as in encoding and transmission of 3D visual data, and even in 3D rendering and interaction.

At a higher level of representation, fully described 3D models may be used for dynamic, real-time generation of stereo content. This will bring more flexibility to the application side enabling implementation of an interactive 3D environment for different view or data manipulation tasks. However, in general, construction of 3D models is computationally expensive and sometimes hard to achieve. Furthermore, the process of projecting the 3D model(s) back to the appropriate 2D (perspective) view(s), namely 3D rendering or 3D visualization, can be computationally expensive. The complexity of the rendering depends on several factors including size of the model(s), *i.e.*, type and number of primitives or constructive elements describing the model, and the desired level of detail and rendering quality. The back-projection process can be even more challenging in the case of (multiview) stereo rendering, where at least two different perspective views are necessary to form virtual stereoscopic 3D space. This increases the difficulty of achieving real-time/interactive frame rates, especially for large models with a great deal of detail or complex scenes composed of several objects. This motivates searching for representations of high performance in terms of memory or processing requirements and accurate and flexible enough to achieve quality rendering and visualization at interactive frame rates, the research topic that we mainly deal with in Part II.

After all, stereo visualization systems add another degree of freedom along the depth dimension. This essentially implies that the current 2D interaction techniques need to be developed for supporting this additional dimension. The upgrade should facilitate free interaction with, and possibly manipulation of, the 3D-visualized objects along the depth, in a manner more or less similar to the other two coordinate components. The users of a 3D system not only look for the common functionalities available on 2D display devices, but they also naturally expect to be able to point

to, and work in, any arbitrary 3D point (voxel) inside the stereoscopic 3D space (as they are able to point to any arbitrary pixel on a 2D display). These are the motivations behind our work in Part III, where we present the results of our experience with a prototype of a stereo 3D cursor and a small set of associated tools we have developed for selection and manipulation of 3D points inside a virtual stereoscopic 3D space.

1.2 Problem Definition, Focus, and Scope

Figure 1.1 shows typical components of a 3D pipeline composed of spatial or spatio-temporal 3D content generation, modeling, possibly transmission (not shown in the figure), visualization, and interaction. As depicted in this figure, our research mainly deals with stereo reconstruction, 3D modeling and visualization, and 3D interaction, which are correspondingly organized in three parts, as follows.

- Part I - Optimal Orthogonal Stereo Sampling for Enhanced 3D Estimation: In this part we study the process of stereo reconstruction from discretized stereo content and show how the 3D estimation can be improved with regard to an optimal trade-off in the density of horizontal versus vertical sampling for a given number of samples (*i.e.* for a given total 2D resolution). More accurately, the focus of the problem is on finding an optimal pixel aspect ratio, or equivalently and optimal horizontal *vs.* vertical discretization, that, given a specific total 2D resolution, improves 3D point estimation from corresponding 2D projections in the left and right stereo views. We study this optimization problem for both parallel and with vergence stereo configurations, and show how the optimal sampling ratio may vary with different underlying stereo parameters. We also extend the research to the case of stereo viewing, where the stereo content is presented to human vision system through a stereoscopic 3D display medium. We propose models of stereo perception on these 3D display devices, based on which we derive a general solution that relates the optimal pixel aspect ratio mainly to the device-specific parameters rather than stereo configuration parameters. This general solution is equally applicable to both stereo-based 3D reconstruction and 3D viewing applications. Our results in this part are based mainly on mathematical modeling and optimization, but we provide substantial numerical results and subjective user studies in support of our theoretical findings.

- Part II - Sampled Object Representation and Visualization: Our research in this part deals mainly with 3D object modeling and representation with the purpose of achieving descriptions that are more efficient in terms of memory requirement, accuracy, and flexibility of supporting different interaction and manipulation tasks at interactive frame rates. Specifically, we will focus on sampled (or point-based) 3D object representations. This is mainly because of the simplicity, flexibility, and performance that these representations demonstrate compared to more traditional representations such as triangular meshes and functional representations. However, these models are usually composed of a huge number of samples and hence need specific treatments in both representation and visualization. In this regard, we will place our focus mainly on hierarchical, multi-resolution representations which efficiently accommodate point-based graphics and have shown great success in 3D rendering, especially in dealing with a large number of primitives [85]. We study these representations and introduce improved techniques, data structures, and algorithms for hierarchical representation and interactive 3D rendering of both still and animated point-based 3D models, and for both conventional and stereoscopic visualization applications.

- Part III - 3D Interaction and Manipulation: This part deals with 3D interaction and manipulation as part of a typical 3D visualization system. Our research in this area is generally connected with the problem of 3D pointing and object selection within a 3D environment, and in particular with the challenges of doing these tasks within a virtual stereoscopic 3D space. In this regard, we study the possibility of using the underlying 3D content, namely point-based representation of the model, in searching and targeting a 3D point. We have developed a prototype of a stereo 3D cursor, in which the underlying content can be used to adjust the depth of the cursor in an automatic, or semi-automatic manner. The cursor is also equipped with a simple set of tools that can be used for simple 3D manipulation tasks within the virtual stereoscopic 3D space. Using this prototype, we study and compare the capabilities of a stereo 3D cursor in stereo space *vs.* its 2D counterpart in conventional 3D rendering on a flat 2D space.

The 3D models that we are using in this study are not necessarily the result of stereo-based 3D reconstruction. Any form of 3D data that can be represented as (or converted to) a collection of 3D points is appropriate for our research. These

include data resulting from 3D laser scanners, data acquired by 3D medical imaging techniques, and synthetic 3D data (see Figure 1.1). As a result, we expect our improvements to be equally beneficial in different application domains such as medical data visualization, computer games, and multimedia education.

1.3 Contributions

We have made several contributions to the field through this research. The first and probably our most significant contribution is the optimal 2D sampling (optimal pixel aspect ratio) model we have proposed for stereo-based 3D reconstruction and enhanced 3D visualization [7, 11] (see Part I). In fact, we propose mathematical models for obtaining the maximum gain out of a given number of samples per 2D view (a given resolution or equivalently a specific bandwidth limit). These optimal sampling models can be of great practical value in the transmission and display of 3D content, especially considering that the 3D displays rely on more than one 2D view (at least two views) of the scene to induce the 3D effect. In this work, we thoroughly reviewed the previous research and unified past work to form a solid, integrated theoretical model. In this regard, we tried to find strong justifications and proofs for different approximations and address unseen cases in the previous related research. We also introduced methods and software for simulating different non-uniform (horizontally, vertically dissimilar) samplings on conventional capture and display devices, which enabled us to run subjective user studies in support of our theoretical findings. We have also extended this work to a general, unified sampling model applicable to the both capturing and viewing ends of the 3D pipeline (see Chapter 4). This latter extension is of great practical value as it facilitates applying the same model in design and manufacturing of the stereo capture and display devices that are used throughout the 3D pipeline.

Our second significant contribution is in the area of point-based representation and rendering, where we introduce a balanced, layered, multi-resolution structure for representation and interactive rendering of large point-based 3D models [10] (see Part II). In this representation, neighboring points are grouped into patches of the same size (same number of samples), and a hierarchy of patches is created for a model. Through this structure we impose a type of regularity on the model without enforcing any explicit connectivity. This regularity facilitates applying/extending conventional 2D encoding techniques (DCT, wavelet, and vector quantization) to the 3D samples and their corresponding attributes, and it also facilitates application

of different deformation models on point-based data. Our representation is very compact compared to similar structures and is also comparable to the other state-of-the-art representations. We have successfully applied our balanced representation in interactive, free view-point rendering of animated point-based 3D models. This extension enables simultaneous spatial and temporal navigation on animated 3D objects within an interactive 3D rendering environment. These generalizations and achievements can be of great practical value in 3D content generation, transmission, and display. As part of this work, a substantial amount of implementation and coding has been done in C/C++ to show the efficiency of the basic idea and its variations, and also to compare the results with other methods proposed in literature.

Our third noteworthy contribution is the stereo 3D cursor that we have developed as a general method for interaction and manipulation within a virtual stereoscopic 3D space (see Part III). In this work, benefiting from the underlying 3D geometry of a scene, we show how easily and conveniently a stereo 3D cursor can be implemented and deployed. Here the initial idea was to investigate the possibility of using stereo visualization techniques in extracting ground truth from 3D medical images [8]. Starting with this idea, we have managed to implement a prototype 3D cursor with some simple associated tools for manipulation of a point-based 3D object within a stereoscopic 3D space. We explain, and also based on user evaluation results show, that the stereo 3D cursor can be used as a general 3D pointing method within such a virtual environment with as many applications that a 2D cursor can have in a conventional 2D interface [9].

1.4 Outline

The rest of this thesis is organized as follows. In Part I we focus on the problem of optimal discretization for stereo reconstruction and stereo-based 3D visualization. This part is composed of four chapters, Chapters 2 to 5, as follows.

In Chapter 2 we review and explain the fundamental concepts of stereopsis process. These include models of 3D estimation from stereo images under two common configurations, namely parallel and with-vergence configurations. We also present a model of 3D viewing on stereoscopic 3D displays and show how this process can also be reduced to (approximated by) a stand-alone (parallel) stereo configuration system. In addition, we discuss the effects of discretization on 3D estimation and 3D (or stereoscopic) resolution. The basic concepts, models, and equations that are presented in this chapter will form the bases of analysis and op-

timizations in the subsequent chapters.

In Chapter 3 we study the problem of optimal 3D point (or 3D scene) estimation from discretized stereo images for a stand-alone stereo system, and for both parallel and with-vergence configurations. Through mathematical modeling and optimization, we drive optimal discretization (optimal pixel aspect ratio) equations for estimating a single 3D point under these configurations. We also present models and equations of optimized discretization for a volume of the scene constrained by the cameras' field of view and a capturing range. In summary, we show that for a given total resolution, a finer horizontal resolution, compared to the usual uniform pixel distribution, in general and under more practical configuration parameters, leads to a more accurate 3D estimation for both configurations, though the optimal value may vary depending on different configuration parameter values.

In Chapter 4 we extend the results derived in Chapter 3 to the case of 3D viewing where the stereo content is presented to the human eyes through a stereoscopic 3D display. In this regard, based on the stereo viewing model presented in Chapter 2, we incorporate the role of the stereoscopy device as a medium into the optimization process and establish formulations which relate the optimal pixel aspect ratio of the 3D display to practical viewing conditions and parameters such as display size and its distance from the human eyes. More importantly, we propose a general mathematical model which is equally applicable to both stereo-based 3D capturing and 3D viewing applications and is independent of the stereo configuration parameters. Our optimization models in this chapter also suggest a finer horizontal *vs.* vertical discretization for a given total resolution. The model, in fact, suggests that vertically rectangular pixels with a ratio of 2:3 (width:height) should be used in the design and manufacture of stereo capturing and stereo display devices.

In Chapter 5 we present our experimental numerical calculations and supportive subjective studies for the theoretical results presented in Chapters 3 and 4. We calculate optimal pixel aspect ratios using the proposed models and equations for some (practical) parameter values. We also explain a method for simulating different non-square pixel aspect ratios on conventional displays. We have used this method to conduct subjective user studies on simulated stereo content of the same resolution but with different pixel aspect ratios. Through these studies we show that the human observers indeed have a better 3D viewing experience with an optimized *vs.* a non-optimized representation of stereo content.

Part II focuses on the sampled (point-based) 3D object representation and ren-

dering. This part includes three chapters, Chapter 6 to Chapter 8, as follows.

In Chapter 6 we briefly review background concepts and related work in point-based 3D object representation and rendering. We review major works in this area, and briefly explain different techniques that are used for the representation and visualization of point-based 3D models. Specifically we discuss hierarchical, multi-resolution representations proposed for interactive rendering of high-volume 3D data. We also discuss the advantages and challenges of creating balanced hierarchies as the basis and initiative of our research and improvements we have made in this area.

In Chapter 7 we introduce a tree structure called a multi-section tree, which allows a multi-way balanced tree to be built for any arbitrary number of samples very close to the structure of a complete tree with the same branching factor. We also introduce algorithms and techniques for implicit representation and traversal of this balanced tree structure. Furthermore, we introduce algorithms that use this tree structure in creating a fully balanced, single- or multi-layer, hierarchical representation of still 3D objects. We also show the results of our algorithms and compare them with some other representations in terms of compactness, rendering quality, and preprocessing complexity.

In Chapter 8 we describe our experience with generalizing the notion of interactive, multi-resolution rendering to the case of sample-based animated 3D objects. Again we use a multi-section tree structure, introduced in Chapter 7, to achieve such an interactive environment over space and time. We introduce techniques and algorithms for hierarchical quantization, encoding, and animated rendering of 4D spatio-temporal samples. Some quantitative and qualitative results on compactness and rendering quality of the representation are also presented at the end.

Part III is devoted to our experience with 3D interaction and 3D manipulation of sampled object representation within a virtual stereoscopic 3D space. This part is composed of two chapters, Chapter 9 and Chapter 10, as follows.

In Chapter 9 we briefly review some basic concepts and techniques of 3D interaction and manipulation. These include the importance of pointing as a prerequisite of performing many tasks, especially within a graphical environment, and the minimum requirements that have to be satisfied by a 3D pointing technique to perform well within a 3D environment.

In Chapter 10 we focus on different aspects of the stereo 3D cursor. We use a multi-view stereo rendering application to evaluate the capability of the 3D versus

2D cursor in manipulating 3D visualized data, and show how the content of a scene can be used to virtually enhance the simplicity and efficiency of object selection and manipulation tasks within a virtual 3D space. We also discuss issues such as pointing accuracy and the problems caused by occlusion, and possible methods of handling these problems.

Finally, Chapter 11 is devoted to a summary of all of the work presented in Chapters 2 to 10, some concluding remarks, and the possible directions in which different parts of this research can be extended.

Part I

Optimal Orthogonal Sampling for Stereo Reconstruction and Visualization

Chapter 2

Stereo-based 3D Reconstruction and Visualization

In this chapter we briefly review some of the basic concepts of stereo reconstruction and 3D visualization. In particular, the geometry of 3D reconstruction from stereo images and the 3D formation process from stereo content presented on stereoscopic displays are explained. We also explain how digitized (discretized) visual content may lead to an intrinsic error, known as *discretization error* [24, 14, 87], in reconstruction of the original 3D scene. These basic concepts and explanations will form the bases of the optimization methods, discussed in the next two chapters, which are aimed to reduce the amount of 3D estimation error originate from discretization.

2.1 Stereo Matching and Stereo 3D Reconstruction

Stereo matching is among the computer vision problems that have been extensively investigated during the past decades [20]. The main purpose of the stereo matching algorithms is to find the correspondence between the 2D locations (2D points or pixels) in two or more views captured of the same scene or object and possibly finding the occluded and non-matched areas in those views. The problem is called *binocular stereo* matching when only two views are used to establish the correspondence; otherwise, it is called *multiview stereo* matching. The results of matching, often called a *disparity map* or a depth map, can then be used in many applications, including recovering the 3D structure of the scene. A number of studies have examined multiview stereo matching in recent years considering that increasing the number of views is generally believed to reduce matching ambiguities, and results

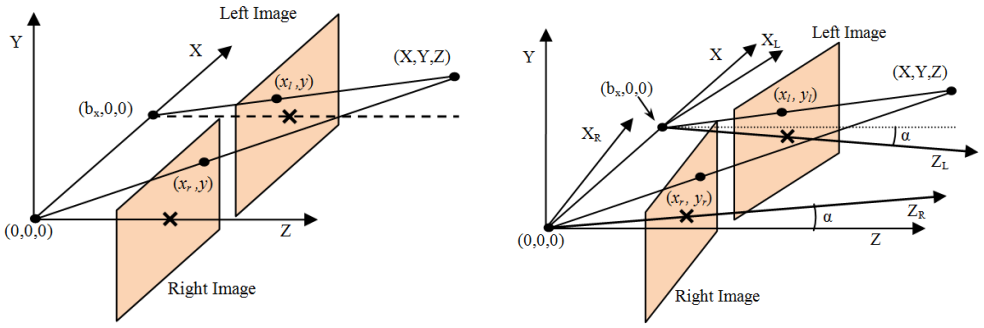
in better, more accurate 3D reconstruction [59]. Multiview stereo can also be used to create a complete 3D model of the object [89, 93]. Depending on the application, the number and direction of views may differ and the views may be captured under controlled or uncontrolled environmental conditions [93].

Establishing correspondence between views, the correspondence problem, is ambiguous for several reasons. These include camera-related issues such as image discretization noise, different camera gains, contrast, and so on; and viewpoint-related problems such as perspective distortions, occlusions, and specular reflections. These issues necessitate applying some additional assumptions or constraints in establishing the correspondence between two (or more) views. Epipolar constraint implies that the matching points lie along conjugate epipolar lines and thus reduces the correspondence problem to a one-dimensional search. The bounded disparity constraint restricts the amount of disparity to a maximum possible value. These two constraints are always valid, and together they greatly reduce the cost and ambiguity of matching. There are also some other constraints that are usually (but not always) valid, yet are applied in many algorithms for simplifying and solving the problem. Uniqueness, ordering, local continuity (smoothness), and same photometric properties are some of these assumptions and constraints [20].

Stereo-matching algorithms include a wide range of different methods, from simple, fast block-matching, which tries to locally find the best matching point by minimizing a criterion such as SAD, sum of absolute differences, in a neighborhood of the point (local optimization), to advanced, often computationally expensive algorithms that try to minimize a global energy function (global optimization) while at the same time imposing a (local) smoothness constraint [20]. Evaluations show that the global optimization methods, in general, generate better results than the local methods, though their performance may notably differ from one image to another [91]. Probabilistic methods [39, 103], such as belief propagation [98] and graph cuts [63], can be considered as examples of global optimization algorithms that show better overall performance in terms of matching accuracy [91, 99].

2.1.1 Geometry of Stereo-based 3D Reconstruction

There are two stereo configurations that are commonly used in stereo capture and stereo-based 3D reconstruction: parallel (Figure 2.1a) and with-vergence (Figure 2.1b) configurations. See also the List of Symbols for a short description of the notation that will be used throughout this thesis.



(a) Parallel stereo

(b) Stereo with vergence

Figure 2.1: Parallel and with-vergence stereo configurations.

Assuming a pinhole camera model [49] and using triangulation rules, the image projections of a 3D point (X, Y, Z) on the left and right image screens in parallel geometry (Figure 2.1a) are given by:

$$x_r = \frac{fX}{Z}, x_l = \frac{f(X - b_x)}{Z}, y = y_r = y_l = \frac{fY}{Z} \tag{2.1}$$

Thus, assuming that corresponding points (x_l, y) and (x_r, y) are known in the left and right image planes, the original 3D point location can be computed as:

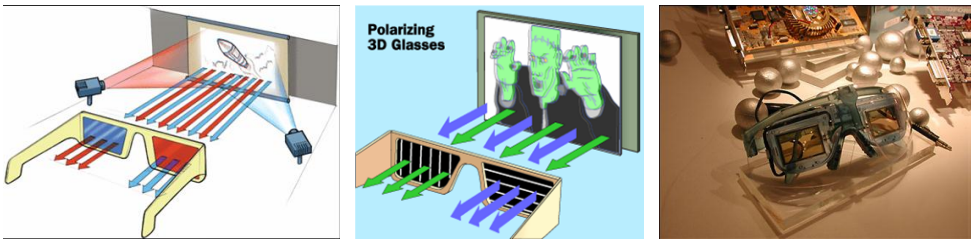
$$X = Z\frac{x_r}{f}, Y = Z\frac{y}{f}, Z = \frac{fb_x}{x_r - x_l} \tag{2.2}$$

For cameras with vergence (Figure 2.1b), the projections of the 3D point (X, Y, Z) on the left and right image planes are given by (see [87] for details):

$$\begin{aligned} x_r &= f\frac{X\cos\alpha - Z\sin\alpha}{Z\cos\alpha + X\sin\alpha} \\ y_r &= f\frac{Y}{Z\cos\alpha + X\sin\alpha} \\ x_l &= f\frac{(X - b_x)\cos\alpha + Z\sin\alpha}{Z\cos\alpha - (X - b_x)\sin\alpha} \\ y_l &= f\frac{Y}{Z\cos\alpha - (X - b_x)\sin\alpha} \end{aligned} \tag{2.3}$$

Again, having the corresponding projections in the left and right images, from Equation 2.3 the original position of the 3D point can be restored as [87]:

$$X = Z\frac{A}{B}, Y = Z\frac{y_r}{B}, Z = b_x\frac{BC}{AC + BD} \tag{2.4}$$



(a) Anaglyph glasses (b) Polarized glasses (c) Shutter glasses

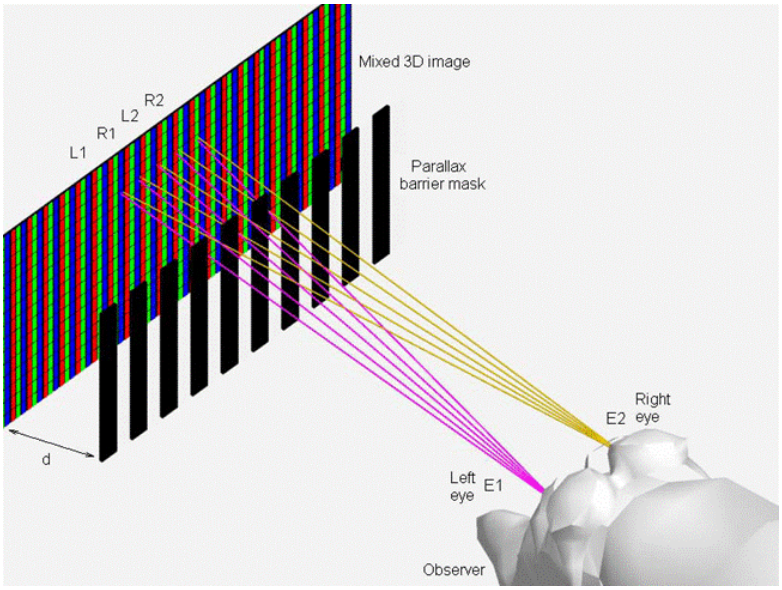
Figure 2.2: Some of common type of glasses that are used to separate the left and right views (Figures from [104]).

where

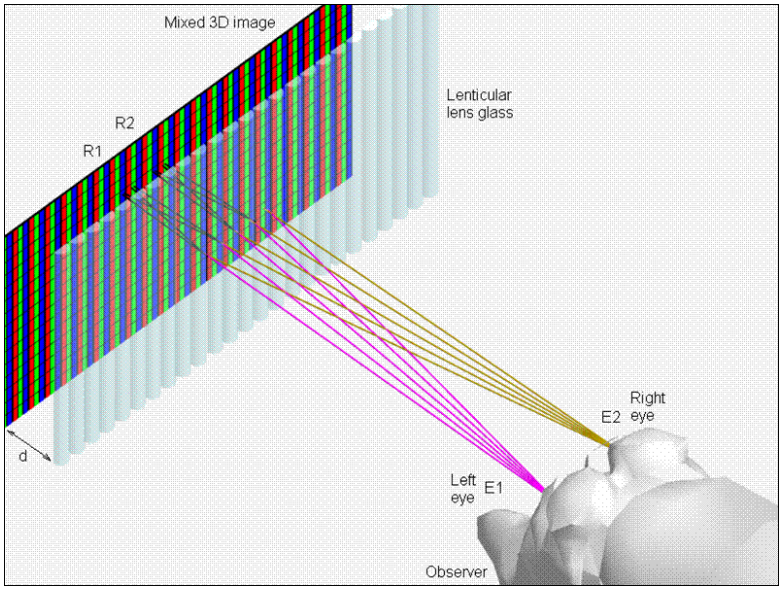
$$A = (f \sin\alpha + x_r \cos\alpha), B = (f \cos\alpha - x_r \sin\alpha)$$
$$C = (f \cos\alpha + x_l \sin\alpha), D = (f \sin\alpha - x_l \cos\alpha)$$

2.2 Watching Stereo Content through a Stereoscopic 3D Display Medium

Stereoscopy is the most common approach of presenting three-dimensional visual content to viewers. These techniques are based on presenting two different views of the scene, captured from (slightly) different perspectives, to the left and right human eyes. The human vision system treats these projections as similar to the images it directly receives from a real scene, and reconstructs a 3D image out of the differences (disparities) in the left and right projections. There are several techniques for separating and conducting the left and right views to the corresponding eyes. A group of these methods use some type of filtering glasses such as color filters (anaglyph Red-Blue or Red-Cyan glasses), polarizing filters, or shutter glasses to separate the left and right views (see Figure 2.2). The drawback of these methods is that the viewers need to wear these glasses when they watch 3D content [104]. The second group of these techniques does not require the use of additional glasses, and therefore, they are called autostereoscopic displays. In these types of systems, the optical aid separating the left and right view for the human eyes is integrated with the display screen. Parallax barriers (Figure 2.3a) and lenticular sheets (Figure 2.3b) are two common types of these optical aides that are used to provide the autostereoscopic capability. Lenticular sheets (optical lens plate) generally provide better 3D picture quality than parallax barriers. Autostereoscopic 3D displays also



(a) Parallax barriers



(b) Lenticular

Figure 2.3: Autostereoscopic 3D devices (Figures from [104]).

provide multiple views of the scene so that the 3D image slightly changes as the observer moves his or her head from one viewing zone to another. Thus, these 3D displays, to some extent, induce the motion-parallax for the observer, as well [104].

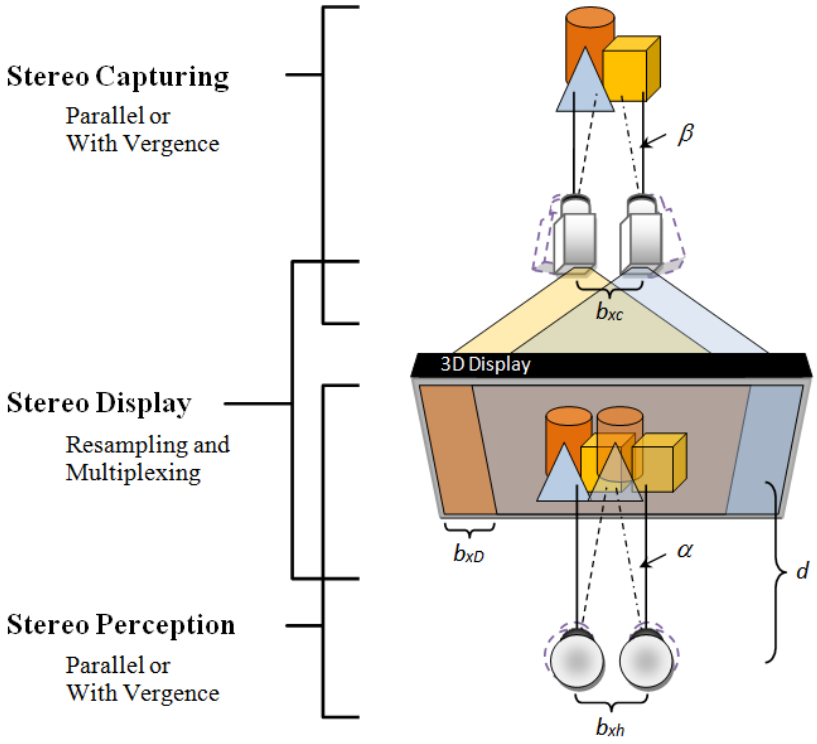


Figure 2.4: The process of stereo capturing, displaying, and perception. The stereo system in both the capturing and the viewing sides can be either in parallel or in with-vergence configuration. Resampling and/or multiplexing may happen when the stereo content is presented on 3D display.

2.2.1 Geometry of 3D Reconstruction through a 3D Display

Section 2.1.1 presents 3D-reconstruction equations for a standalone stereo system, whereas, as illustrated in Figure 2.4, two stereo systems are involved in the process of stereo perception through a stereoscopic 3D display:

1. a real/virtual stereo capturing/rendering system that is used to provide the stereo content, and

2. the human stereo vision system, which tries to reconstruct the 3D scene from the stereo content provided.

In this process, the stereoscopy device, either in the form of glasses or head-mounted displays, or integrated with the display system, has the role of a medium that directs the left/right views to the corresponding human eyes.

As depicted in in Figure 2.4, stereo systems in both the capturing and the viewing sides can be either in parallel or in with-vergence configuration, independent of the other one. Thus, four different scenarios are possible for the whole process,

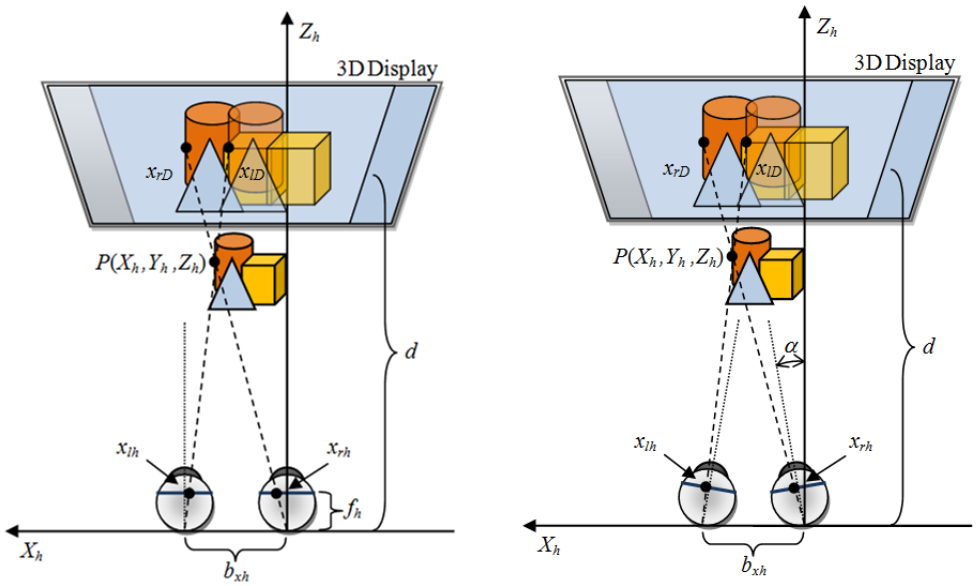


Figure 2.5: Watching stereo images captured under parallel configuration through a 3D display device: (a) assuming parallel configuration for human eyes, and (b) assuming vergence configuration for human eyes.

which will be discussed and modeled in this section. We will use subscripts c, D, and h to refer to different components of the process: c for a stereo imaging system (cameras), D for a stereoscopic 3D display, and h for the human vision system. For example f_c stands for the focal length of the cameras while f_h denotes the eyes' focal length (see also List of Symbols).

If we consider parallel geometry for the capturing side, then according to Equation 2.1, the projection of a 3D point (X, Y, Z) on left and right camera image planes is given by:

$$x_{rc} = \frac{f_c X}{Z}, x_{lc} = \frac{f_c (X - b_{xc})}{Z}, y_c = y_{rc} = y_{lc} = \frac{f_c Y}{Z} \tag{2.5}$$

The images captured by the stereo cameras may be scaled by a factor S, *i.e.* re-sampled, and possibly multiplexed before presenting on the stereoscopic display device. Thus, the corresponding 2D point coordinates in display space are given by:

$$x_{rD} = S x_{rc}, x_{lD} = S x_{lc}, y_D = y_{rD} = y_{lD} = S y_c \tag{2.6}$$

Now, if we assume that display is placed at distance d of the human eyes, and the sight direction is perpendicular to the display screen, then the 3D points that are

perceived by the left and right human eyes will take the following form:

$$\begin{aligned} P_l &= (x_{lD}, y_D, d) \\ P_r &= (x_{rD}, y_D, d) \end{aligned} \tag{2.7}$$

Now, it is not difficult to show that in this case, in which the stereo content is captured under parallel configuration, the original 3D scene can be viewed without distortion up to a scaling factor and irrespective of the human stereo configuration. If we assume parallel configuration for human eyes (Figure 2.5a), the projection of P_l and P_r on the corresponding eye is obtained, using Equation 2.1, as follows [7]:

$$x_{rh} = \frac{f_h x_{rD}}{d}, x_{lh} = \frac{f_h x_{lD}}{d}, y_h = \frac{f_h y_D}{d} \tag{2.8}$$

From Equation 2.8 the 3D point $P(X_h, Y_h, Z_h)$ reconstructed by human eyes is given theoretically by:

$$\begin{aligned} X_h &= Z_h \frac{x_{rh}}{f_h} = Z_h \frac{x_{rD}}{d} \\ Y_h &= Z_h \frac{y_h}{f_h} = Z_h \frac{y_D}{d} \\ Z_h &= \frac{f_h b_{xh}}{x_{rh} - x_{lh}} = \frac{d b_{xh}}{x_{rD} - x_{lD}} \end{aligned} \tag{2.9}$$

If we assume that human eyes watch stereo content with a vergence angle α (Figure 2.5b), then using Equation 2.3 the corresponding point projections on human eyes can be calculated as:

$$\begin{aligned} x_{rh} &= f_h \frac{x_{rD}\cos\alpha - d\sin\alpha}{d\cos\alpha + x_{rD}\sin\alpha} \\ y_{rh} &= f_h \frac{y_D}{x_{rD}\sin\alpha + d\cos\alpha} \\ x_{lh} &= f_h \frac{x_{lD}\cos\alpha + d\sin\alpha}{d\cos\alpha - x_{ld}\sin\alpha} \\ y_{lh} &= f_h \frac{y_D}{x_{lD}\sin\alpha + d\cos\alpha} \end{aligned} \tag{2.10}$$

From the above equations the reconstructed 3D point $P(X_h, Y_h, Z_h)$ for this case is obtained as:

$$X_h = Z_h \frac{A}{B}, Y_h = Z_h \frac{y_{rh}}{B}, Z_h = b_{xh} \frac{BC}{AC + BD} \tag{2.11}$$

where

$$A = (f_h \sin\alpha + x_{rh}\cos\alpha), B = (f_h \cos\alpha - x_{rh}\sin\alpha)$$
$$C = (f_h \cos\alpha + x_{lh}\sin\alpha), D = (f_h \sin\alpha - x_{lh}\cos\alpha)$$

Substituting Equation 2.10 into Equation 2.11, Equation 2.9 is obtained again. For instance, by substituting x_{rh} into X_h equation we have:

$$\begin{aligned}
X_h &= Z_h \frac{f_h \sin\alpha + f_h \frac{x_{rD}\cos\alpha - d\sin\alpha}{d\cos\alpha + x_{rD}\sin\alpha}\cos\alpha}{f_h \cos\alpha - f_h \frac{x_{rD}\cos\alpha - d\sin\alpha}{d\cos\alpha + x_{rD}\sin\alpha}\sin\alpha} \\
&= Z_h \frac{d\sin\alpha\cos\alpha + x_{rD}\sin\alpha^2 + x_{rD}\cos\alpha^2 - d\sin\alpha\cos\alpha}{d\cos\alpha^2 + x_{rD}\sin\alpha\cos\alpha - x_{rD}\sin\alpha\cos\alpha + d\sin\alpha^2} \\
&= Z_h \frac{x_{rD}(\sin\alpha^2 + \cos\alpha^2)}{d(\sin\alpha^2 + \cos\alpha^2)} \\
&= Z_h \frac{x_{rD}}{d}
\end{aligned}$$

Derivations for Y_h and Z_h are similar.

This means that, in theory, the 3D-scene reconstructed by the human eyes from stereo images that are captured/rendered under parallel configuration is independent of the amount of the vergence of the human eyes.

If stereo content is captured under vergence, it is still possible to configure the display device so that the original 3D scene can be reconstructed without distortion. For this purpose, the left and right images should be inwardly rotated to duplicate the capturing vergence angle, and the human eyes should be positioned at a specific distance so that the original epipolar planes can be resembled. In practice, a specific viewing distance cannot be enforced in many applications. Moreover, current 3D displays do not have any mechanism for supporting vergence. As a result, the stereo images that are captured with some vergence may lead to distorted or deformed 3D scenes when presented to the viewer [56]. In this case, the stereo images can be rectified before presentation on a 3D display [49]. However, for small vergence angles we may assume that the human eyes compensate for the inconsistent vertical disparities to establish correspondence between the left and right projections [7]. As a result, Equation 2.9 can be used as a general model for 3D estimation by human eyes when the stereo content is viewed using a stereoscopic 3D display device. In other words, the 3D reconstruction via a 3D display is also reduced to a single stereo system where the display viewing distance d and the baseline of the human visual system b_{xh} are the focal length and baseline of the system, respectively, and the left

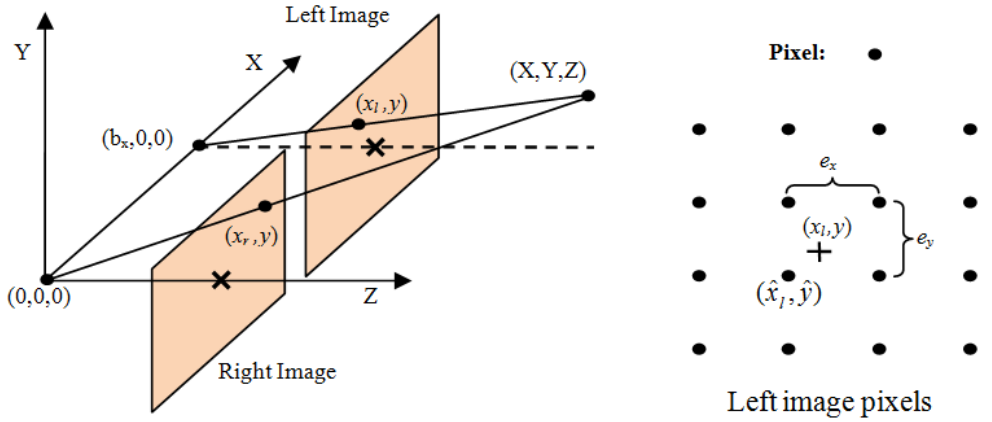


Figure 2.6: Left: Parallel-camera stereo projection with baseline b_x. Right: Discretization error.

and right display screens serve as its left and right image planes.

Note that, for the sake of simplicity, we have dropped the subscript c in all equations referring to a standalone stereo capturing configuration throughout this thesis, unless it is not clear enough from the context that we are talking about a single stereo capturing system. Subscripts D and h still will be used to discriminate between different spaces and components throughout the remainder of this thesis.

2.3 Discretization Error in Stereo Reconstruction and Visualization

In the world of digital imaging and digital display devices, (stereo) images are represented in the form of a grid of (generally square) pixels. In this representation the actual point projections (x, y) are rounded to the nearest pixel position $(\hat{x}, \hat{y})$ (see Figure 2.8). Apart from the effects of this discretization in finding the corresponding projections, this introduces an inherent error in 3D estimation called *discretization error* [14, 24, 87]. In fact, since the location of a 3D-point $P(X, Y, Z)$ is determined using rounded point projections rather than the actual ones, in the case of parallel configuration (Figure 2.6) the location of P is estimated based on $(\hat{x_r}, \hat{y})$ and $(\hat{x_l}, \hat{y})$ as:

$$\hat{X} = \hat{Z}\frac{\hat{x_r}}{f}, \hat{Y} = \hat{Z}\frac{\hat{y}}{f}, \hat{Z} = \frac{fb_x}{\hat{x_r} - \hat{x_l}} \tag{2.12}$$

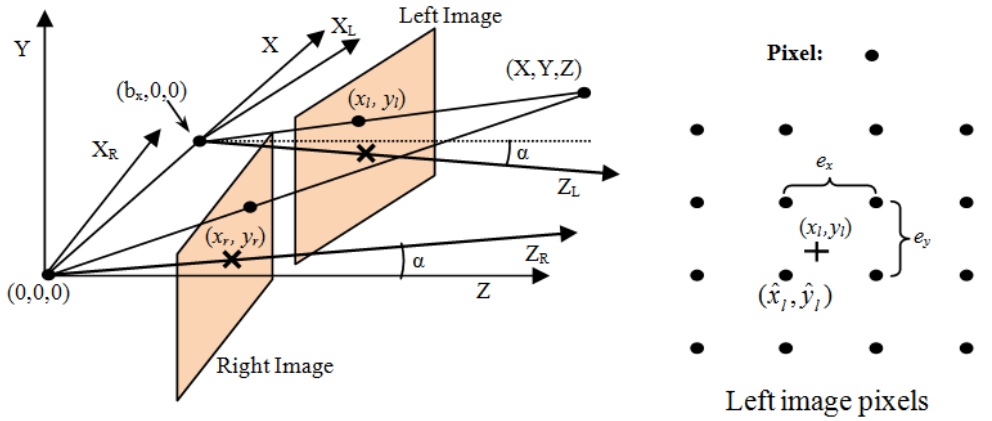


Figure 2.7: Left: Stereo camera projection with vergence α and baseline b_x. Right: Discretization error.

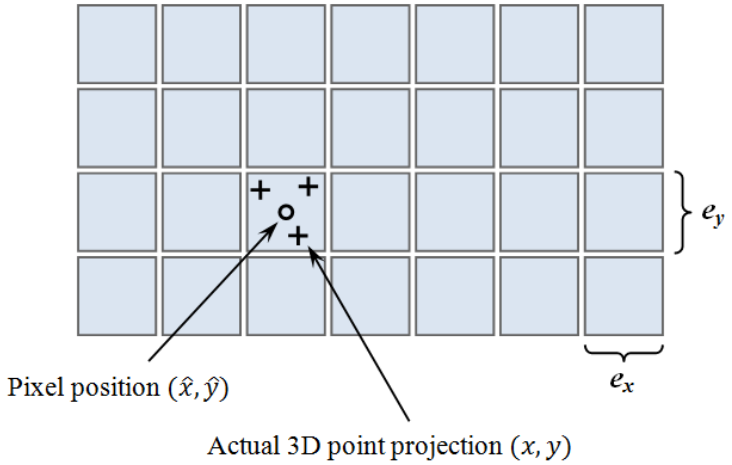
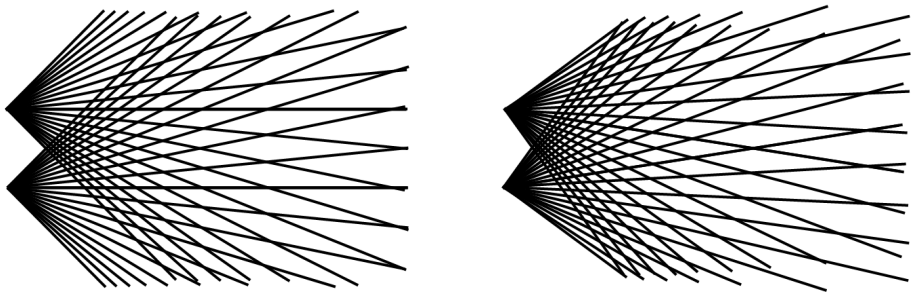


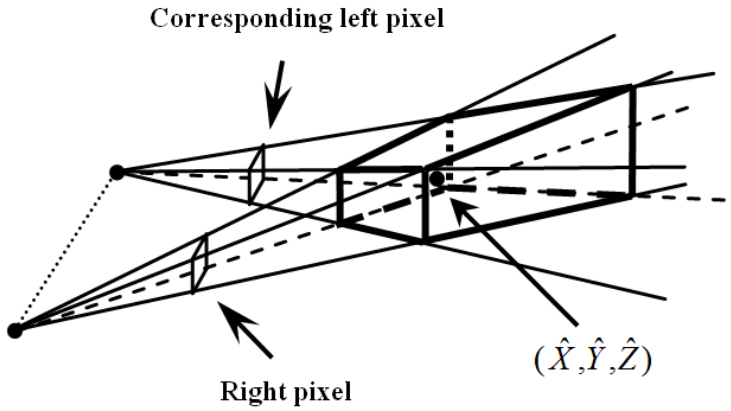
Figure 2.8: Rounding off of the actual 3D point projections to the nearest pixel due to the discrete nature of digital imaging or display devices.

Similarly, in the case of stereo with vergence (Figure 2.7), because of the discretization error $(\hat{x_r}, \hat{y_r})$ and $(\hat{x_l}, \hat{y_l})$ will be used in the computations resulting in an estimation error with respect to the original 3D point.

Discretization is closely related to the *3D resolution* or *stereoscopic resolution* which is defined in terms of the precision in locating 3D points within the comfortable viewing range of a 3D display [52] (see Figure 2.10). The magnitude of the discretization error (or the coarseness of the 3D resolution) depends on the pixel distance (pixel size) in 2D views and is proportional to the viewing distance. Figure 2.9 shows the error pattern for two camera configurations: parallel geometry



(a) (b)



(c)

Figure 2.9: Discretization error patterns: (a) parallel-axes stereo, (b) stereo with vergence, and (c) 3D-view of error pattern for two corresponding pixels.

(Figure 2.9a) and with-vergence geometry (Figure 2.9b). Figure 2.9c illustrates a 3D-view of an error generated by two corresponding pixels. The quadrilateral volume in Figure 2.9c corresponds to a voxel (3D pixel) of a stereoscopic 3D display or a 3D region that is projected on a pair of corresponding pixels.

Obviously, reducing the pixel size and thus increasing the total resolution will reduce size of the voxels (3D pixels) or equivalently will reduce the amount of 3D point estimation error. However, assuming a constant total resolution R, it is reasonable to look for an optimal pixel aspect ratio which minimizes the size of voxels or equivalently the discretization error inside the viewing region. Such optimization can improve the output image quality of a typical 3D TV system while the size of the input data is unchanged. The optimal solution can also be beneficial for stereo-based depth estimation and 3D reconstruction problems [20, 91].

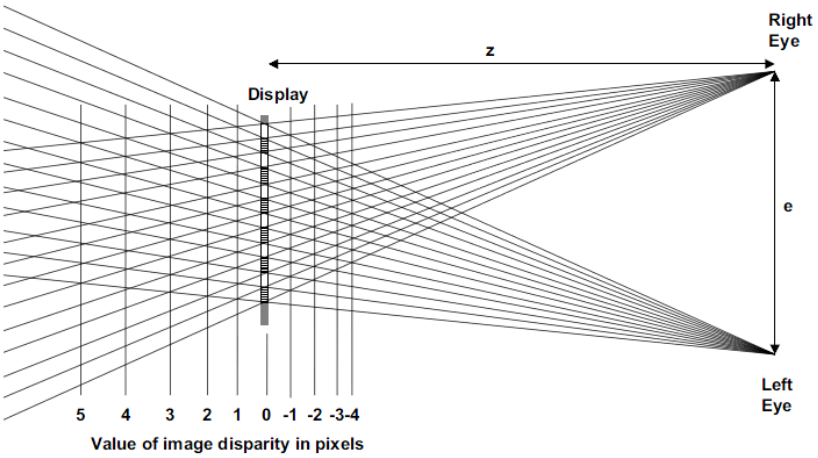


Figure 2.10: Stereoscopic resolution of 3D displays (Figure from [52]).

2.4 Related Work on Optimal Discretization

An early investigation on the optimal discretization problem, as stated above, dates back to the work of Basu [14] with the aim of using the results to improve the 3D-point estimation in stereo reconstruction applications. The approach is based on establishing a tight upper bound on the relative estimation error on the Y component, and then minimizing this error function to achieve the best horizontal versus vertical discretization with regard to the maximum relative error in the estimation of Y and for a single 3D point. This result then is extended to the case of optimization over a capturing volume constrained by a range of depth values, Z_{min} to Z_{max}, and a range of y values, $-y_{max}$ to $+y_{max}$, in order to derive a solution of more practical value. The study is limited to the parallel stereo configuration. Following a similar approach, Cheng *et al.* have extended these results to a capturing region enclosed by the field of view of the cameras and the afore-mentioned depth and height ranges for both parallel and with vergence stereo configurations [24, 27].

In a different approach to the problem Sahabi and Basu [15, 87] have studied the effect of vergence, the inward or outward turning of the cameras to look at an object of interest, on the depth estimation error in two different case. First, they consider a uniform pixel distribution over the x and y axes of the cameras' image plane. They show that in this case the vergence can be used to reduce the depth estimation error, but this error cannot be minimized without some adjustments in the cameras' focal length. In the second case, they consider a spatially varying pixel distribution similar to the human eyes where the resolution is more dense in

the center of the image plane and (exponentially) reduces toward the peripheries. In this case, the optimal vergence angle, *i.e.*, the vergence angle that minimizes the object depth estimation error, is the angle at which both cameras look at the object of interest [87].

Extensive research has also been conducted on stereoscopic 3D imaging techniques and 3D display design and technologies [56, 16, 75, 74]. Specifically, research shows that wider display screens with aspect ratios (display width:display height) between 5:3 and 6:3 are more visually pleasant to most viewers [73, 76] and provide a better sensation of depth even with 2D images (display aspect ratio should not be confused with pixel aspect ratio, which is the focus of this research). This observations have led to the design of wider screens for HDTV displays than those used in previous TV generations [56]. As another closely related study, Konrad *et al.* [64] analyze and model the process of view subsampling and multiplexing in lenticular autostereoscopic 3D displays. This analysis is used to derive the specifications of some optimal filter to reduce the aliasing effect. When deployed prior to view multiplexing, these optimal anti-aliasing filters demonstrate improvements in perceived 3D quality for this class of 3D displays and generally for 3D display technologies that are based on spatial view multiplexing.

In the next three chapters, we study the optimization problem elucidated above for both stereo configurations, parallel and with vergence. We review and integrate previous work [14, 24, 27], and provide comprehensive proofs and justifications for previous equations and results, and also some extensions to the case of stereo with vergence [27]. We also extend the research to the case of stereo viewing, where the stereo content is presented to the human vision system through a stereoscopic 3D display device or medium. In this regard, we propose a general model that relates the optimal pixel aspect ratio mainly to the device-specific parameters rather than stereo configuration parameters, and hence is equally applicable to both stereo-based 3D reconstruction and 3D viewing applications. To support these theoretical findings, we also present extensive numerical results and subjective user studies for the optimal solutions obtained for these different models and scenarios.

Chapter 3

Optimal Sampling for Stereo Reconstruction

In this chapter we study the problem of optimal 3D point (or 3D scene) estimation from discretized stereo images. Reviewing and extending the earlier research ([14, 87, 15, 24, 27]), we show how the 3D estimation can be improved with regard to the density of horizontal versus vertical sampling for a given number of samples (total resolution) and for different stereo configurations: parallel and with vergence. We also show how the optimal sampling ratio (pixel aspect ratio) varies with different configuration parameters.

3.1 Optimal 3D Estimation for a Single 3D point

As mentioned earlier (see Section 2.3), the discrete nature of a digital imaging device leads to rounding error. The rounding error is bounded by half a pixel (see Figure 2.8). Thus, in the worst case:

$$\begin{aligned} \hat{x_r} &= x_r \pm (e_x/2) \\ \hat{x_l} &= x_l \pm (e_x/2) \\ \hat{y_r} &= y_r \pm (e_y/2) \\ \hat{y_l} &= y_l \pm (e_y/2) \end{aligned} \tag{3.1}$$

Considering the stereo setup with parallel configuration (Figure 2.6) where $y_r = y_l = y$, and the worst-case error in 3D estimation, which will be our default as-

sumption in the rest of this part, from Equations 2.12 and 3.1 $\hat{Z}$ is obtained as [14]:

$$\hat{Z} = \frac{fb_x}{(x_r - x_l) \pm e_x} = Z\left(1 \pm e_x \frac{Z}{fb_x}\right)^{-1} \tag{3.2}$$

The Taylor series expansion of Equation 3.2 about $Z = 0$ is given by:

$$\begin{aligned}\hat{Z} &= Z\left(1 \pm e_x \frac{Z}{fb_x}\right)^{-1} \\ &= Z\left(1 \pm \frac{e_x}{fb_x}Z \pm \left(\frac{e_x}{fb_x}\right)^2 Z^2 \pm \left(\frac{e_x}{fb_x}\right)^3 Z^3 + \cdots\right)\end{aligned} \tag{3.3}$$

In practice f and b_x values are much greater than values taken by e_x. Therefore, we can assume that the higher order terms in the above expansion are negligible. As a result, $\hat{Z}$ can be properly approximated as:

$$\hat{Z} \cong Z\left(1 \pm e_x \frac{Z}{fb_x}\right) \tag{3.4}$$

Bounds on error in estimating X can be obtained as follows:

$$\begin{aligned}\hat{X} = \frac{\hat{Z}}{f}\hat{x_r} &\cong \frac{Z}{f}\left(1 \pm \frac{e_x Z}{fb_x}\right)\left(x_r \pm \frac{e_x}{2}\right) \\ &= X\left(1 \pm \frac{e_x Z}{fb_x} \pm \frac{e_x}{2x_r} \pm \frac{e_x^2 Z}{2fb_x x_r}\right)\end{aligned} \tag{3.5}$$

Similarly, for Y:

$$\hat{Y} = Y\left(1 \pm \frac{e_x Z}{fb_x} \pm \frac{e_y}{2y} \pm \frac{e_x e_y Z}{2fb_x y}\right) \tag{3.6}$$

or,

$$\left|\frac{\hat{Y} - Y}{Y}\right| \leq g_p(e_x, e_y) = \left\{\frac{e_x Z}{fb_x} + \frac{e_y}{2|y|} + \frac{e_x e_y Z}{2f|y|b_x}\right\} \tag{3.7}$$

Considering a unit image capturing or viewing area:

$$\left(\frac{1}{e_x}\right)\left(\frac{1}{e_y}\right) = R \text{ or } e_y = \frac{1}{e_x R} \tag{3.8}$$

Applying Equation 3.8 in 3.7 we have:

$$g_p(e_x) = \left\{\left(\frac{Z}{fb_x}\right)e_x + \left(\frac{1}{2R|y|}\right)\frac{1}{e_x} + \frac{Z}{2f|y|Rb_x}\right\} \tag{3.9}$$

From Equation 3.5 it is obvious that the best solution which minimizes the estimation error of X or Z is to have e_x as small as possible. Nevertheless, this is also the worst possible choice for estimating the Y component. As a compromise, Equation 3.9, which is the upper bound of the relative estimation error in Y as a function of e_x constrained by the total resolution R through Equation 3.8, can be used to find the best vertical *vs.* horizontal resolution trade-off for estimating Y. In fact, optimizing for the maximum relative error in estimation of Y implicitly involves X and Z components while at the same time prevents significant degradation in estimation of the Y component itself. This motivates the derivations of a mathematical model for optimal discretization for a single 3D point in parallel configuration as follows [14]:

Result 1. *Considering the parallel stereo configuration in Figure 2.6, the optimal discretization for estimating Y for a single 3D point (X, Y, Z) is:*

$$e_x = \frac{1}{\sqrt{R}}\sqrt{\frac{fb_x}{2|y|Z}},\ e_y = \frac{1}{\sqrt{R}}\sqrt{\frac{2|y|Z}{fb_x}} \tag{3.10}$$

Proof. The results are obtained by equating the derivative of $g_p(e_x)$ in Equation 3.9 to zero to minimize the error:

$$g'_p(e_x) = \left\{ \frac{Z}{fb_x} - \frac{1}{e_x^2}\left(\frac{1}{2R|y|}\right)\right\} \tag{3.11}$$

Solving the above equation in terms of e_x gives the value of e_x in Equation 3.10. To show that the function g_p gets the minimum value at this point we need to perform the second derivative test. From the above equation the second derivative of g_p in terms of e_x is calculated as:

$$g''_p(e_x) = \left(\frac{1}{R|y|}\right)\frac{1}{e_x^3} \tag{3.12}$$

On the other hand, from Equation 3.8 we can see that e_x changes over the open interval $(0, \infty)$ which means that e_x always takes positive values. Since the other two parameters involved in the above equation are also positive, the second derivative of g_p is always positive over the domain of possible values for e_x. As a result, g_p is strongly convex over this domain and takes its minimum value at e_x given by Equation 3.10. The value of e_y in Equation 3.10 can be obtained from e_x by applying Equation 3.8. □

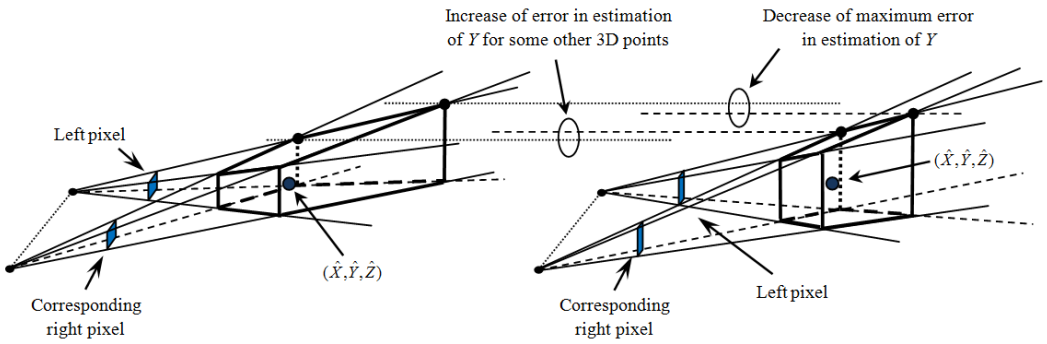


Figure 3.1: Comparing discretization error pattern and changes of error in estimation of Y for regular square pixels, $e_x = e_y$, (left) *vs.* vertically rectangular pixels, $e_x < e_y$, with optimal pixel aspect ratio (right).

It is worth mentioning that the above solution is also optimal for absolute estimation error on Y and estimation error on Y relative to $|YZ|$, which are respectively defined as follows:

$$\left|Y - \hat{Y}\right| \leq \frac{Z}{f}\left(\left(\frac{Z|y|}{fb_x}\right)e_x + \left(\frac{1}{2R}\right)\frac{1}{e_x} + \frac{Z}{2Rfb_x}\right) \tag{3.13}$$

$$\left|\frac{Y - \hat{Y}}{YZ}\right| \leq \left(\frac{1}{fb_x}\right)e_x + \left(\frac{1}{2R|y|Z}\right)\frac{1}{e_x} + \frac{1}{2Rf|y|b_x} \tag{3.14}$$

This can be proven in the same way as Result 1. This means that the optimal solution is not affected by the monotonic increase of error along with the Y and Z components, which in turn allows the discretization optimization problem to be studied independent of the pixel location on the image plane.

The solution that is obtained in this way is not only optimal in terms of the maximum (relative) estimation error on Y, but often yields better estimates for the X and Z components, as well. From 3.10 the ratio of e_x to e_y can be calculated as:

$$\frac{e_x}{e_y} = \frac{fb_x}{2|y|Z} = \frac{(x_r - x_l)}{2|y|} \tag{3.15}$$

In Equation 3.15, $(x_r - x_l)$ is the amount of the disparity between the corresponding points in the left and right image planes and is always greater than or equal to zero (note that x_r and x_l are measured with respect to the corresponding camera coordinate system). The amount of disparity $(x_r - x_l)$ in most (practical) cases is smaller than $2|y|$. This means that in practice the ratio in Equation 3.15 is usually less than

one, or equivalently e_x is often smaller than e_y. As a result, the optimal solution in Result 1, while minimizing the maximum error in estimation of Y, often improves the estimates of the X and Z components as well. However, when $e_x < e_y$ the optimal solution changes the error pattern so that the Y estimation error may be increased for some 3D points estimated by $(\hat{X}, \hat{Y}, \hat{Z})$. This phenomenon is pictorially illustrated in Figure 3.1 for a typical error pattern. As shown in this figure, the optimal pixel aspect ratio less than one (*i.e.* $e_x < e_y$) results in improvement in estimation of X and Z plus reduction of the maximum error in estimation of Y, which in turn means improvement in the estimation of Y for some 3D points. But generally this is achieved in the cost of degradation in the estimation of Y for some other 3D points.

Now we focus on the stereo setup with vergence, shown in Figure 2.7. We start with some auxiliary derivations as follows.

Auxiliary Result 1. *Assuming a small vergence angle α, the estimated depth $\hat{Z}$ of a 3D point for the vergence configuration (Figure 2.7) can be related to the true depth Z by:*

$$\hat{Z} \cong Z \left(1 \pm \frac{e_x Z \cos\alpha}{2b_x(f\cos\alpha - x_r \sin\alpha)} \pm \frac{e_x Z \cos\alpha}{2b_x(f\cos\alpha + x_l \sin\alpha)} \right)^{-1} \tag{3.16}$$

Proof. From Equation 2.4, Z is estimated as:

$$\hat{Z} = \frac{b_x(f\cos\alpha + \hat{x_l}\sin\alpha)(f\cos\alpha - \hat{x_r}\sin\alpha)}{(f\cos\alpha + \hat{x_l}\sin\alpha)(f\sin\alpha + \hat{x_r}\cos\alpha) + (f\sin\alpha - \hat{x_l}\cos\alpha)(f\cos\alpha - \hat{x_r}\sin\alpha)} \tag{3.17}$$

Using the same approach as the case of without vergence:

$$\begin{aligned} \hat{Z} &= \left(\frac{f\sin\alpha + \hat{x_r}\cos\alpha}{b_x(f\cos\alpha - \hat{x_r}\sin\alpha)} + \frac{f\sin\alpha - \hat{x_l}\cos\alpha}{b_x(f\cos\alpha + \hat{x_l}\sin\alpha)} \right)^{-1} \\ &= \left(\frac{f\sin\alpha + (x_r \pm (e_x/2))\cos\alpha}{b_x(f\cos\alpha - (x_r \pm (e_x/2))\sin\alpha)} + \frac{f\sin\alpha - (x_l \pm (e_x/2))\cos\alpha}{b_x(f\cos\alpha + (x_l \pm (e_x/2))\sin\alpha)} \right)^{-1} \end{aligned} \tag{3.18}$$

Assuming that $(e_x/2)\sin\alpha \approx 0$ for a small vergence angle α:

$$\begin{aligned}\hat{Z} &\cong \left(\frac{f\sin\alpha + 2x_r\cos\alpha \pm e_x\cos\alpha}{2b_x(f\cos\alpha - x_r\sin\alpha)} + \frac{2f\sin\alpha - 2x_l\cos\alpha \pm e_x\cos\alpha}{2b_x(f\cos\alpha + x_l\sin\alpha)}\right)^{-1} \\ &= \left(Z^{-1} \pm \frac{e_x\cos\alpha}{2b_x(f\cos\alpha - x_r\sin\alpha)} \pm \frac{e_x\cos\alpha}{2b_x(f\cos\alpha + x_l\sin\alpha)}\right)^{-1} \\ &= Z\left(1 \pm \frac{e_xZ\cos\alpha}{2b_x(f\cos\alpha - x_r\sin\alpha)} \pm \frac{e_xZ\cos\alpha}{2b_x(f\cos\alpha + x_l\sin\alpha)}\right)^{-1} \end{aligned} \tag{3.19}$$

□

Auxiliary Result 2. *Assuming a small vergence angle α, for the vergence configuration the estimated depth of a 3D point is bounded by:*

$$\hat{Z} \leq Z\left(1 + \frac{e_xZ}{fb_x}K\right) \textit{where} K = \left(1 + \frac{x_{max}}{f}\tan\alpha\right) \tag{3.20}$$

Proof. As in the parallel configuration case (Equation 3.3) the Taylor expansion of Equation 3.16 about $Z = 0$ can be calculated. Assuming a small vergence angle α, the higher order terms in the Taylor series expansion can be ignored, and $\hat{Z}$ can be properly approximated as:

$$\hat{Z} \cong Z\left(1 \pm \frac{e_xZ}{2b_x}\frac{\cos\alpha}{(f\cos\alpha - x_r\sin\alpha)} \pm \frac{e_xZ}{2b_x}\frac{\cos\alpha}{(f\cos\alpha + x_l\sin\alpha)}\right) \tag{3.21}$$

Expanding the second term of Equation 3.21 and ignoring higher order terms, we have:

$$\frac{\cos\alpha}{(f\cos\alpha - x_r\sin\alpha)} = \frac{1}{f}\left(1 - \frac{x_r\sin\alpha}{f\cos\alpha}\right)^{-1} \approx \frac{1}{f}\left(1 + \frac{x_r}{f}\tan\alpha\right) \tag{3.22}$$

Similarly, for the third term:

$$\frac{\cos\alpha}{(f\cos\alpha + x_l\sin\alpha)} = \frac{1}{f}\left(1 + \frac{x_l\sin\alpha}{f\cos\alpha}\right)^{-1} \approx \frac{1}{f}\left(1 - \frac{x_l}{f}\tan\alpha\right) \tag{3.23}$$

From Equations 3.21, 3.22, and 3.23:

$$\begin{aligned}\hat{Z} &\approx Z\left(1 \pm \frac{e_x Z}{2b_x}\frac{1}{f}\left(1 + \frac{x_r}{f}\tan\alpha\right) \pm \frac{e_x Z}{2b_x}\frac{1}{f}\left(1 - \frac{x_l}{f}\tan\alpha\right)\right) \\ &= Z\left(1 \pm \frac{e_x Z}{2fb_x}\left(2 + \frac{(x_r - x_l)}{f}\tan\alpha\right)\right) \\ &\leq Z\left(1 + \frac{e_x Z}{fb_x}\left(1 + \frac{x_{max}}{f}\tan\alpha\right)\right) \end{aligned} \tag{3.24}$$

where $x_{max} = \max\left(\frac{x_r - x_l}{2}\right)$ is half of the maximum possible horizontal disparity, which is in fact equal to the maximum possible value of x or equivalently half of the image plane width. □

Auxiliary Result 3. *Assuming a small vergence angle α:*

$$\left|\frac{\hat{Y} - Y}{Y}\right| \leq g_v(e_x) = \left\{\left(\frac{ZK}{fb_x}\right)e_x + \left(\frac{1}{2R|y_r|}\right)\frac{1}{e_x} + \frac{ZK}{2f|y_r|Rb_x}\right\} \tag{3.25}$$

Proof. For a small vergence angle α, the parallel configuration equations can be used to approximate the Y component. Therefore:

$$\begin{aligned}\hat{Y} \approx \frac{\hat{Z}\hat{y_r}}{f} &\leq \frac{Z}{f}\left(1 + \frac{e_x Z}{fb_x}K\right)\left(y_r \pm \frac{e_y}{2}\right) \\ &\approx Y\left(1 + \frac{e_x ZK}{fb_x} \pm \frac{e_y}{2y_r} \pm \frac{e_x e_y ZK}{2fb_x y_l}\right)\end{aligned} \tag{3.26}$$

Considering a total resolution of R, Equation 3.25 can be obtained from Equations 3.8 and 3.26. □

Summarizing the above-mentioned derivations, the following result can be stated for a stereo configuration with vergence.

Result 2. *Assuming a stereo system with a small vergence angle α (Figure 2.7), the optimal discretization for a single 3D point can be obtained as:*

$$e_x = \frac{1}{\sqrt{R}}\sqrt{\frac{fb_x}{2|y_r|ZK}}, \quad e_y = \frac{1}{\sqrt{R}}\sqrt{\frac{2|y_r|ZK}{fb_x}} \tag{3.27}$$

Proof. As in the parallel configuration, from Equation 3.25:

$$g'_v(e_x) = \left\{\frac{ZK}{fb_x} - \left(\frac{1}{2R|y_r|}\right)\frac{1}{e_x^2}\right\} \tag{3.28}$$

The results can be obtained by equating Equation 3.28 to zero, and solving it in terms of e_x. The same argument as Result 1 can be applied to show that g_v indeed takes its minimum value at this point, considering the domain of permissible values for e_x. □

3.2 Optimizing within a Viewing Volume

The derivations in the last section consider optimization with respect to a single 3D point. Instead, we need to consider a region in 3D defined by a set of constraints on the range of values in depth, height and the stereo Field-Of-View (FOV). Obviously, the particular values obtained for e_x (or e_y) in the previous section will not be optimum for all points satisfying these constraints. A solution to this problem could be calculating the integral of the error function g_p or g_v over the entire specified region. However, this approach leads to logarithmic terms which complicate subsequent steps in the optimization process. Instead, we consider minimizing an appropriate error metric subject to the restrictions imposed.

For the parallel camera configuration (Figure 2.6), we use Equation 3.11 to obtain such an error metric. In this case, from Equation 3.11 for the optimal discretization e_x for a single 3D point we have:

$$\frac{Z}{fb_x} = \frac{1}{e_x^2}\left(\frac{1}{2R|y|}\right)$$
$$\frac{fb_x}{Z} = |y|2e_x^2R$$

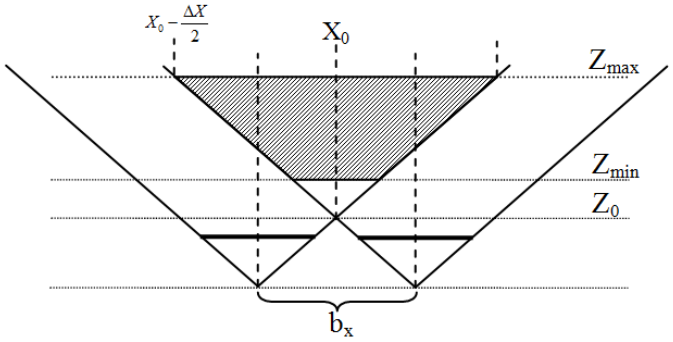
or equivalently

$$\frac{fb_x}{Z} - |y|2e_x^2R = 0 \tag{3.29}$$

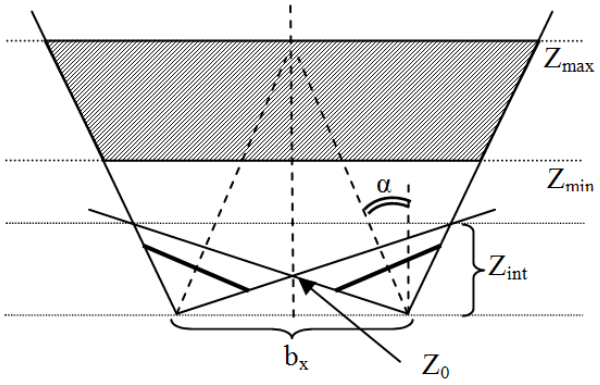
In other words, for a single 3D point, the derivative of g_p or equivalently the right side of Equation 3.29 is zero at the optimal discretization and non-zero in its neighbourhood. This, in fact, gives a distance criterion that can be used as an error metric for optimizing the discretization error over the entire region. We define the square of this distance as the error metric, as follows:

$$E_p = \left(\frac{fb_x}{Z} - |y|2e_x^2R\right)^2 \tag{3.30}$$

On the other hand, for a parallel configuration the range of values of X varies with changes of depth Z and values of the cameras' field of view according to the



(a) Parallel configuration



(b) With-vergence configuration

Figure 3.2: Viewing region constrained by a range of values on depth (Z_{min}, Z_{max}) and field of view of the cameras.

following simple relationship (see Figure 3.2a):

$$X \in \left\{ X_0 \pm \frac{\Delta X}{2} \frac{(Z - Z_0)}{(Z_{max} - Z_0)} \right\} \tag{3.31}$$

Thus, we need to optimize the following function w.r.t. e_x:

$$G_p(e_x) = \int_{Z_{min}}^{Z_{max}} \int_{-y_{max}}^{+y_{max}} \int_{X_0 - \frac{\Delta X}{2}\frac{(Z-Z_0)}{(Z_{max}-Z_0)}}^{X_0 + \frac{\Delta X}{2}\frac{(Z-Z_0)}{(Z_{max}-Z_0)}} \left(\frac{f b_x}{Z} - |y| 2 e_x^2 R \right)^2 dX dy dZ \tag{3.32}$$

Calculating the integral in 3.32 and equating its derivative to zero gives us the following result:

Result 3. *Considering the parallel stereo configuration given in Figure 3.2a, the*

discretization in x for optimizing the average error in estimation of Y is given by:

$$e_x = \left(-\frac{fb_x}{2R} \frac{Z_0 I_1[Z_{min}, Z_{max}] - I_2[Z_{min}, Z_{max}]}{I_3[Z_{min}, Z_{max}] - Z_0 I_4[Z_{min}, Z_{max}]} \right)^{\frac{1}{2}} \tag{3.33}$$

where,

$$\begin{aligned}
I_1[Z_1, Z_2] &= \int_{Z_1}^{Z_2} \int_{-y_{max}}^{+y_{max}} \frac{|y|}{Z} dy dZ = y_{max}^2 \ln\left(\frac{Z_2}{Z_1}\right) \\
I_2[Z_1, Z_2] &= \int_{Z_1}^{Z_2} \int_{-y_{max}}^{+y_{max}} |y| dy dZ = y_{max}^2 (Z_2 - Z_1) \\
I_3[Z_1, Z_2] &= \int_{Z_1}^{Z_2} \int_{-y_{max}}^{+y_{max}} Z y^2 dy dZ = \frac{y_{max}^3}{3} (Z_2^2 - Z_1^2) \\
I_4[Z_1, Z_2] &= \int_{Z_1}^{Z_2} \int_{-y_{max}}^{+y_{max}} y^2 dy dZ = \frac{2y_{max}^3}{3} (Z_2 - Z_1)
\end{aligned} \tag{3.34}$$

Proof.

$$\begin{aligned}
G_p(e_x) &= \int_{Z_{min}}^{Z_{max}} \int_{-y_{max}}^{+y_{max}} \int_{X_0 - \frac{\Delta X}{2} \frac{(Z-Z_0)}{(Z_{max}-Z_0)}}^{X_0 + \frac{\Delta X}{2} \frac{(Z-Z_0)}{(Z_{max}-Z_0)}} \left(\frac{fb_x}{Z} - |y| 2e_x^2 R \right)^2 dX dy dZ \\
&= \int_{Z_{min}}^{Z_{max}} \int_{-y_{max}}^{+y_{max}} \Delta X \frac{(Z - Z_0)}{(Z_{max} - Z_0)} \left(\frac{f^2 b_x^2}{Z^2} - 4Re_x^2 fb_x \frac{|y|}{Z} + 4R^2 e_x^4 y^2 \right) dy dZ \\
&= e_x^2 \left[\frac{4\Delta X Z_0 R f b_x}{(Z_{max} - Z_0)} \int_{Z_{min}}^{Z_{max}} \int_{-y_{max}}^{+y_{max}} \frac{|y|}{Z} dy dZ - \frac{4\Delta X R f b_x}{(Z_{max} - Z_0)} \int_{Z_{min}}^{Z_{max}} \int_{-y_{max}}^{+y_{max}} |y| dy dZ \right] \\
&+ e_x^2 \left[\frac{4\Delta X Z_0 R f b_x}{(Z_{max} - Z_0)} \int_{Z_{min}}^{Z_{max}} \int_{-y_{max}}^{+y_{max}} \frac{|y|}{Z} dy dZ - \frac{4\Delta X R f b_x}{(Z_{max} - Z_0)} \int_{Z_{min}}^{Z_{max}} \int_{-y_{max}}^{+y_{max}} |y| dy dZ \right] \\
&+ I_0
\end{aligned} \tag{3.35}$$

where I_0 is independent of e_x, and thus has no influence on the minimization of

$G_p(e_x)$.

$$\begin{aligned} G_p(e_x) = e_x^2 & \left[\frac{4\Delta X Z_0 R f b_x}{Z_{max} - Z_0} I_1[Z_{min}, Z_{max}] - \frac{4\Delta X R f b_x}{Z_{max} - Z_0} I_2[Z_{min}, Z_{max}] \right] \\ & + e_x^4 \left[\frac{4\Delta X R^2}{Z_{max} - Z_0} I_3[Z_{min}, Z_{max}] - \frac{4\Delta X R^2 Z_0}{Z_{max} - Z_0} I_4[Z_{min}, Z_{max}] \right] \\ & + I_0 \end{aligned} \tag{3.36}$$

where I_1 to I_4 are the aforementioned integrals.

Considering the derivative of $G_p(e_x)$ in terms of e_x and equating it to zero, we have:

$$\begin{aligned} G_p'(e_x) = 2e_x & \left[\frac{4\Delta X Z_0 R f b_x}{Z_{max} - Z_0} I_1[Z_{min}, Z_{max}] - \frac{4\Delta X R f b_x}{Z_{max} - Z_0} I_2[Z_{min}, Z_{max}] \right] \\ & + 4e_x^3 \left[\frac{4\Delta X R^2}{Z_{max} - Z_0} I_3[Z_{min}, Z_{max}] - \frac{4\Delta X R^2 Z_0}{Z_{max} - Z_0} I_4[Z_{min}, Z_{max}] \right] \\ & = 0 \end{aligned} \tag{3.37}$$

Since we cannot take $e_x = 0$ as the optimal point, we should have:

$$\begin{aligned} & f b_x (Z_0 I_1[Z_{min}, Z_{max}] - I_2[Z_{min}, Z_{max}]) \\ & + 2R e_x^2 (I_3[Z_{min}, Z_{max}] - Z_0 I_4[Z_{min}, Z_{max}]) = 0 \end{aligned} \tag{3.38}$$

Solving this equation in terms of e_x, and again considering that we cannot take negative values for e_x as the optimal point, we obtain the following equation (or Equation 3.33) as the possible optimal point:

$$e_x = \left(-\frac{f b_x}{2R} \frac{Z_0 I_1[Z_{min}, Z_{max}] - I_2[Z_{min}, Z_{max}]}{I_3[Z_{min}, Z_{max}] - Z_0 I_4[Z_{min}, Z_{max}]} \right)^{\frac{1}{2}} = e_{xOpt}$$

For ease of reference in the rest of the argument, we have renamed the value of e_x in the above equation as e_{xOpt}. Now, we need to show that $G_p(e_{xOpt})$ is the only minimum value of $G_p(e_x)$ over the domain of possible values for e_x (as we discussed earlier in Result 1, the domain of possible values for e_x is the open interval $(0, \infty)$). For this purpose, we first show that $G_p(e_x)$ has a local minimum

at e_{xOpt} under the specified assumptions, and then we show that it does not take any other minimum over the $(0, \infty)$ interval. We use the second derivative test to show that $G_p(e_x)$ has a local minimum at e_{xOpt}:

$$\begin{aligned} G_p''(e_x) = \frac{8\Delta XR}{Z_{max} - Z_0} [fb_x(Z_0 I_1[Z_{min}, Z_{max}] - I_2[Z_{min}, Z_{max}]) \\ + 6Re_x^2(I_3[Z_{min}, Z_{max}] - Z_0 I_4[Z_{min}, Z_{max}])] \end{aligned} \tag{3.39}$$

Therefore,

$$\begin{aligned} G_p''(e_{xOpt}) &= \frac{8\Delta XR}{Z_{max} - Z_0} [fb_x(Z_0 I_1[Z_{min}, Z_{max}] - I_2[Z_{min}, Z_{max}]) \\ &+ 6R\left(-\frac{fb_x}{2R}\right)\left(\frac{Z_0 I_1[Z_{min}, Z_{max}] - I_2[Z_{min}, Z_{max}]}{I_3[Z_{min}, Z_{max}] - Z_0 I_4[Z_{min}, Z_{max}])}\right) \\ &\times (I_3[Z_{min}, Z_{max}] - Z_0 I_4[Z_{min}, Z_{max}])] \\ &= \frac{8\Delta XR}{Z_{max} - Z_0} [fb_x(Z_0 I_1[Z_{min}, Z_{max}] - I_2[Z_{min}, Z_{max}]) \\ &- 3fb_x(Z_0 I_1[Z_{min}, Z_{max}] - I_2[Z_{min}, Z_{max}])] \\ &= \frac{8\Delta XR}{Z_{max} - Z_0}[2fb_x(I_2[Z_{min}, Z_{max}] - Z_0 I_1[Z_{min}, Z_{max}])] \end{aligned} \tag{3.40}$$

Replacing I_1 and I_2 with their definitions we have:

$$G_p''(e_{xOpt}) = \left(\frac{8\Delta XR}{(Z_{max} - Z_0)} fb_x y_{max}^2\right)((Z_{max} - Z_{min}) - Z_0(\ln Z_{max} - \ln Z_{min}))$$

If $G_p''(e_{xOpt}) > 0$ then $G_p(e_x)$ has a local minimum at e_{xOpt}. Since the first factor in the above equation is positive, this condition implies that the second factor should be positive. In other words, we should have:

$$(Z_{max} - Z_{min}) - Z_0(\ln Z_{max} - \ln Z_{min}) > 0 \tag{3.41}$$

or equivalently,

$$\frac{Z_{max} - Z_{min}}{\ln Z_{max} - \ln Z_{min}} > Z_0$$

Considering that $Z_{max} \geq Z_{min}$, we can assume $Z_{max} = rZ_{min}, r \geq 1$. Substituting

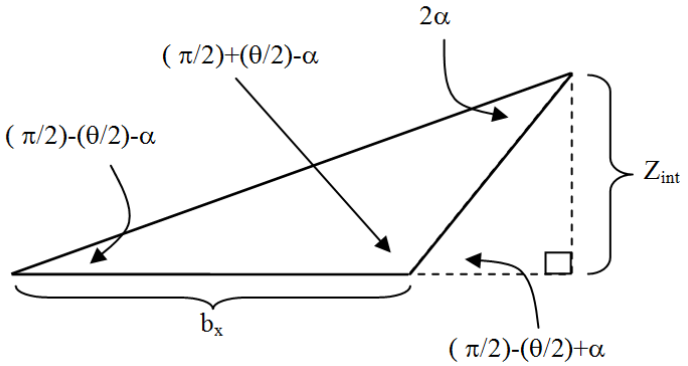


Figure 3.3: A close-up of Figure 3.2b illustrating its geometric constraints.

this in the above equation we have:

$$\frac{rZ_{min} - Z_{min}}{\ln(rZ_{min}) - \ln Z_{min}} > Z_0$$
$$\frac{Z_{min}(r-1)}{\ln r} > Z_0$$
$$\frac{Z_{min}}{Z_0} \times \frac{(r-1)}{\ln r} > 1$$

Now considering that $\frac{Z_{min}}{Z_0} \geq 1$, based on the above equation we have

$$\frac{r-1}{\ln r} \geq 1$$

This last equation is always true for $r \geq 1$, and as a result Equation 3.41 is true. Therefore $G_p(e_{xOpt})$ is a local minimum.

Following the same procedure as above for the other two extrema, *i.e.*, $e_x = 0$ and $e_x = -e_{xOpt}$, it is not difficult to show that $G_p(0)$ and $G_p(-e_{xOpt})$ are the local maximum and local minimum, respectively. Therefore, considering that $G_p(-e_{xOpt})$, $G_p(0)$, and $G_p(e_{xOpt})$ are the only local extrema of function $G_p(e_x)$, $G_p(e_{xOpt})$ should be the only minimum, or equivalently a global minimum, over open interval $(0, \infty)$. □

Determining the range of values of X when Z varies over a specific domain is a little more complicated for the vergence configuration (see Figure 3.2b). This is because between Z_0 to Z_{int} the values of X vary in a certain way, and for $Z > Z_{int}$ the values of X vary in a different way. The values of Z_0 and Z_{int} are given through the following equations.

Auxiliary Result 4. *Given a field of view θ and a vergence angle α:*

$$Z_0 = \frac{b_x}{2}\tan\left(\frac{\pi}{2} - \alpha - \frac{\theta}{2}\right)$$
$$Z_{int} = \frac{b_x}{2}\frac{\cos(2\alpha) + \cos\theta}{\sin(2\alpha)} \tag{3.42}$$

Proof. Follows from Figures 3.2 and 3.3, using simple geometric rules, such as the sine rule. The details are omitted. □

Auxiliary Result 5. *Considering the setup with vergence in Figure 3.2b, the range of values of X, depending on depth Z and the field of view of the cameras, varies based on the following rules:*

$$\Delta X_{int} = \frac{b_x}{2}\frac{\sin\theta}{\sin(2\alpha)} \quad \textit{for} \quad Z = Z_{int} \tag{3.43}$$

$$\Delta X = \Delta X_{int}\frac{(Z - Z_0)}{(Z_{int} - Z_0)} \quad \textit{for} \quad Z_0 < Z < Z_{int} \tag{3.44}$$

$$\Delta X = (\Delta X_{int} - b_x)\frac{Z}{Z_{int}} + b_x \quad \textit{for} \quad Z_{int} \leq Z \tag{3.45}$$

(It is assumed that the X-axis is the dashed line joining the centers of the two cameras.)

Proof. Follows from trigonometric rules and the earlier derivations. Details are skipped here. □

Here again, we need to consider an appropriate error metric to minimize over the range of possible X, Y, and Z values. Similar to the parallel configuration scenario, we use the following error metric based on the derivative in Equation 3.28:

$$E_v = \left(\frac{fb_x}{Z} - |y_r|2RKe_x^2\right)^2 \tag{3.46}$$

We thus need to optimize the following function w.r.t. e_x (from this point forward, for simplicity we ignore subscript r in y_r):

$$G_v(e_x) = \int_Z \int_y \int_X \left(\frac{fb_x}{Z} - |y_r|2RKe_x^2\right)^2 dXdydZ \tag{3.47}$$

Unlike the multiple integral in Equation 3.32, the computation of the above integral may need to be divided into two parts depending on the values of Z_{min} and

Z_{max} relative to Z_{int}. If $Z_{int} \leq Z_{min}$ or $Z_{max} \leq Z_{int}$, we need to consider either Equation 3.44 or Equation 3.45 in computing this integral. However, if $Z_{min} < Z_{int} < Z_{max}$, then we must consider both Equations 3.44 and 3.45.

By calculating $G_v(e_x)$ in these three possible cases and equating corresponding derivatives to zero, we can obtain optimal discretization for all cases. Thus, we can state the following general result.

Result 4. *Considering the stereo system framework in Figure 3.2b, the discretization in x for optimizing the average error in estimation of Y is given by:*

I) $Z_{int} \leq Z_{min}$

$$e_x = \left(\frac{fb_x}{2RK} \frac{A[Z_{min}, Z_{max}]}{B[Z_{min}, Z_{max}]} \right)^{\frac{1}{2}} \tag{3.48}$$

II) $Z_{max} \leq Z_{int}$

$$e_x = \left(\frac{fb_x}{2RK} \frac{C[Z_{min}, Z_{max}]}{D[Z_{min}, Z_{max}]} \right)^{\frac{1}{2}} \tag{3.49}$$

III) $Z_{min} < Z_{int} < Z_{max}$

$$e_x = \left(\frac{fb_x}{2RK} \frac{A[Z_{min}, Z_{int}] + C[Z_{int}, Z_{max}]}{B[Z_{min}, Z_{int}] + D[Z_{int}, Z_{max}]} \right)^{\frac{1}{2}} \tag{3.50}$$

where,

$$\begin{aligned}
A[Z_1, Z_2] &= b_x I_1[Z_1, Z_2] + \frac{(\Delta X_{int} - b_x)}{Z_{int}} I_2[Z_1, Z_2] \\
B[Z_1, Z_2] &= b_x I_4[Z_1, Z_2] + \frac{(\Delta X_{int} - b_x)}{Z_{int}} I_3[Z_1, Z_2] \\
C[Z_1, Z_2] &= I_2[Z_1, Z_2] - Z_0 I_1[Z_1, Z_2] \\
D[Z_1, Z_2] &= I_3[Z_1, Z_2] - Z_0 I_4[Z_1, Z_2]
\end{aligned} \tag{3.51}$$

I_1 to I_4 are equations defined in Result 3.

Proof.

Case I) $Z_{int} \leq Z_{min}$

Considering the equations in Auxiliary Result 5, the integral in Equation 3.47

should be calculated according to the following range of values:

$$G_v(e_x) = \int_{Z_{min}}^{Z_{max}} \int_{-y_{max}}^{+y_{max}} \int_{-\frac{1}{2}\left[(\Delta X_{int}-b_x)\frac{Z}{Z_{int}}+b_x\right]}^{+\frac{1}{2}\left[(\Delta X_{int}-b_x)\frac{Z}{Z_{int}}+b_x\right]} \left(\frac{fb_x}{Z} - 2RKe_x^2|y|\right)^2 dXdydZ$$

Thus,

$$\begin{aligned}
G_v(e_x) = & \int_{Z_{min}}^{Z_{max}} \int_{-y_{max}}^{+y_{max}} \left((\Delta X_{int} - b_x)\frac{Z}{Z_{int}} + b_x\right) \\
& \left(\frac{f^2b_x^2}{Z^2} - 4RKe_x^2 fb_x\frac{|y|}{Z} + 4R^2K^2e_x^4y^2\right) dydZ \\
& = e_x^2\left[-4RKfb_x^2 I_1[Z_{min}, Z_{max}] - \frac{4(\Delta X_{int} - b_x)RKfb_x}{Z_{int}} I_2[Z_{min}, Z_{max}]\right] \\
& + e_x^4\left[\frac{4(\Delta X_{int} - b_x)R^2K^2}{Z_{int}} I_3[Z_{min}, Z_{max}] + 4R^2K^2b_x I_4[Z_{min}, Z_{max}]\right] \\
& + I_0
\end{aligned}$$

where I_0 is again independent of e_x and has no effect on the optimization of $G_v(e_x)$, and I_1 to I_4 are defined as in Result 3.

$$\begin{aligned}
G_v'(e_x) = & 2e_x\left[-4RKfb_x^2 I_1[Z_{min}, Z_{max}] - \frac{4(\Delta X_{int} - b_x)RKfb_x}{Z_{int}} I_2[Z_{min}, Z_{max}]\right] \\
& + 4e_x^3\left[\frac{4(\Delta X_{int} - b_x)R^2K^2}{Z_{int}} I_3[Z_{min}, Z_{max}] + 4R^2K^2b_x I_4 I_4[Z_{min}, Z_{max}]\right] \\
& = 0
\end{aligned}$$

or

$$\begin{aligned}
& \left[-fb_x^2 I_1[Z_{min}, Z_{max}] - \frac{(\Delta X_{int} - b_x)fb_x}{Z_{int}} I_2[Z_{min}, Z_{max}]\right] \\
& + 2e_x^2\left[\frac{4(\Delta X_{int} - b_x)RK}{Z_{int}} I_3[Z_{min}, Z_{max}] + RKb_x I_4[Z_{min}, Z_{max}]\right] \\
& = 0
\end{aligned}$$

Solving the last equation in terms of e_x and again considering that we cannot pick negative values as the optimal point, the value given in Equation 3.48 is obtained. To show that the function $G_v(e_x)$ takes a minimum value at this point and that this is

its global minimum value over the allowable values for e_x, we calculate the second derivative of $G_p(e_x)$ at this point:

$$\begin{aligned}
G_v''(e_x) &= 2\left[-4RKfb_x^2 I_1[Z_{min}, Z_{max}] - \frac{4(\Delta X_{int} - b_x)RKfb_x}{Z_{int}} I_2[Z_{min}, Z_{max}]\right] \\
&+ 12e_x^2\left[\frac{4(\Delta X_{int} - b_x)R^2K^2}{Z_{int}} I_3[Z_{min}, Z_{max}] + 4R^2K^2 b_x I_4 I_4[Z_{min}, Z_{max}]\right] \\
&= 8RK\left[-fb_x\left(b_x I_1[Z_{min}, Z_{max}] + \frac{\Delta X_{int} - b_x}{Z_{int}} I_2[Z_{min}, Z_{max}]\right)\right. \\
&\left.+12e_x^2 RK\left(\frac{\Delta X_{int} - b_x}{Z_{int}} I_3[Z_{min}, Z_{max}] + b_x I_4[Z_{min}, Z_{max}]\right)\right]
\end{aligned}$$

Recalling the expression in Equation 3.48 as e_{xOpt}, the value of second derivative of $G_v(e_x)$ at e_{xOpt} is calculated as:

$$\begin{aligned}
G_v''(e_{xOpt}) &= 8RK\left[-fb_x\left(b_x I_1[Z_{min}, Z_{max}] + \frac{\Delta X_{int} - b_x}{Z_{int}} I_2[Z_{min}, Z_{max}]\right)\right. \\
&+ 12RK\left(\frac{fb_x}{2RK}\right)\left(\frac{b_x I_1[Z_{min}, Z_{max}] + \frac{\Delta X_{int} - b_x}{Z_{int}} I_2[Z_{min}, Z_{max}]}{\frac{\Delta X_{int} - b_x}{Z_{int}} I_3[Z_{min}, Z_{max}] + b_x I_4[Z_{min}, Z_{max}]}\right) \\
&\left.\left(\frac{\Delta X_{int} - b_x}{Z_{int}} I_3[Z_{min}, Z_{max}] + b_x I_4[Z_{min}, Z_{max}]\right)\right] \\
&= 40RKfb_x\left(b_x I_1[Z_{min}, Z_{max}] + \frac{\Delta X_{int} - b_x}{Z_{int}} I_2[Z_{min}, Z_{max}]\right)
\end{aligned}$$

Considering the definitions of I_1 and I_2 and also noting that $(\Delta X_{int} - b_x)$ is greater than zero, we can see that $G_v''(e_{xOpt})$ is greater than zero. As a result, $G_v(e_{xOpt})$ is a local minimum for $G_v(e_x)$. Following the same procedure for the other two extrema, it is not difficult to show that $G_v(0)$ and $G_v(-e_{xOpt})$ are local maximum and local maximum, respectively. Since $G_v(-e_{xOpt})$, $G_v(0)$, and $G_v(e_{xOpt})$ are the only extrema of $G_v(e_x)$, we can say that $G_v(e_{xOpt})$ is the global minimum over the interval $(0, \infty)$.

Case II) $Z_{max} \leq Z_{int}$

In this case following integral should be calculated:

$$G_v(e_x) = \int_{Z_{min}}^{Z_{max}} \int_{-y_{max}}^{+y_{max}} \int_{-\frac{1}{2}\left[\Delta X_{int}\frac{Z-Z_0}{Z_{int}-Z_0}\right]}^{+\frac{1}{2}\left[\Delta X_{int}\frac{Z-Z_0}{Z_{int}-Z_0}\right]} \left(\frac{fb_x}{Z} - 2RKe_x^2|y|\right)^2 dX dy dZ$$

$$= \int_{Z_{min}}^{Z_{max}} \int_{-y_{max}}^{+y_{max}} \left(\Delta X_{int}\frac{Z-Z_0}{Z_{int}-Z_0}\right)\left(\frac{f^2b_x^2}{Z^2} - 4RKe_x^2 fb_x\frac{|y|}{Z} + 4R^2K^2e_x^4y^2\right) dy dZ$$

As in Case I, calculating the derivative of $G_v(e_x)$ and equating it to zero gives the result. To show that the expression obtained, *i.e.* Equation 3.49, minimizes G_v over the domain of allowable values for e_x, the same argument as in Result 3 should be applied. Because of the many similarities with the derivation of Result 3 the details are skipped here.

Case III) $Z_{min} < Z_{int} < Z_{max}$
In this case the integral in Equation 3.47 should be divided into two parts as follows.

$$G_v(e_x) = \int_{Z_{min}}^{Z_{max}} \int_{-y_{max}}^{+y_{max}} \int_{-\frac{1}{2}\left[\Delta X_{int}\frac{Z-Z_0}{Z_{int}-Z_0}\right]}^{+\frac{1}{2}\left[\Delta X_{int}\frac{Z-Z_0}{Z_{int}-Z_0}\right]} \left(\frac{fb_x}{Z} - 2RKe_x^2|y|\right)^2 dX dy dZ$$

$$+ \int_{Z_{min}}^{Z_{max}} \int_{-y_{max}}^{+y_{max}} \int_{-\frac{1}{2}\left[(\Delta X_{int}-b_x)\frac{Z}{Z_{int}}+b_x\right]}^{+\frac{1}{2}\left[(\Delta X_{int}-b_x)\frac{Z}{Z_{int}}+b_x\right]} \left(\frac{fb_x}{Z} - 2RKe_x^2|y|\right)^2 dX dy dZ$$

The other steps are similar to those in Cases I and II and are skipped here. □

Chapter 4

Optimal Sampling for Stereo-based 3D Visualization

Chapter 3 explains the process of optimizing 3D estimation from stereo images for a standalone (parallel or with-vergence) stereo configuration. In stereo-based 3D visualization, the stereo content is presented to the human eyes through a stereoscopy device such as an autostereoscopic 3D display, stereo filtering glasses or a head-mounted display. In this regard, it is necessary to consider the role of the stereoscopy device as a medium, and the effect of other parameters such as viewing distance, as well as the actual behavior of the eyes when they are watching a 3DTV, in the optimization process. In this chapter we use the stereo viewing model suggested in Section 2.2 to incorporate these factors into the optimization process and establish formulations which relate the optimal PAR of the 3D display to practical viewing conditions and parameters such as display size and its distance from the human eyes. More importantly, we extend our research [7] by proposing a general mathematical model which is applicable to both stereo-based 3D reconstruction and 3D viewing applications. This is achieved by transferring the optimization problem from the stereoscopic 3D space to the image plane/display screen space. The solution that is obtained in this way is related mainly to the image plane/3D display aspect ratio, a device-specific parameter, rather than to the stereo configuration parameters. This unification, as well as its independence of the stereo parameters, is practically very important as it facilitates applying the same discretization model to both the capturing and the viewing ends of the 3D pipeline.

4.1 3D Estimation Through a Stereo-based 3D Display Medium

As explained in Section 2.2.1 the whole process of 3D viewing via a 3D display system can be reduced to a single stereo system. In this system, the focal length is the viewing distance to the 3D display, d, the baseline is the baseline of the human visual system, b_{xh}, and the left and right display screens play the role of its left and right image planes (see Figure 2.5). In this system the 3D point that is reconstructed by human eyes can be obtained using Equation 2.9, which is repeated here for ease of reference:

$$\begin{aligned} X_h &= Z_h \frac{x_{rh}}{f_h} = Z_h \frac{x_{rD}}{d} \\ Y_h &= Z_h \frac{y_h}{f_h} = Z_h \frac{y_D}{d} \\ Z_h &= \frac{f_h b_{xh}}{x_{rh} - x_{lh}} = \frac{d b_{xh}}{x_{rD} - x_{lD}} \end{aligned} \tag{4.1}$$

As in the case of digital stereo images, due to the discretized nature of the display screen the actual projections are rounded off to the nearest pixel. Therefore, the location of a 3D-point is determined using $(\hat{x}_{rD}, \hat{y}_D)$ and $(\hat{x}_{lD}, \hat{y}_D)$. This implies that the 3D-point (X_h, Y_h, Z_h) is estimated as:

$$\hat{X}_h = \hat{Z}_h \tfrac{\hat{x}_{rD}}{d}, \quad \hat{Y}_h = \hat{Z}_h \tfrac{\hat{y}_D}{d}, \quad \hat{Z}_h = \tfrac{d b_{xh}}{(\hat{x}_{rD} - \hat{x}_{lD})} \tag{4.2}$$

Following the same procedure as in the case of direct estimation from stereo images (Section 3.1), and again considering the worst-case error in 3D estimation, bounds on error in estimating Y_h can be obtained as follows:

$$\left| \frac{\hat{Y}_h - Y_h}{Y_h Z_h} \right| \leq f(e_x, e_y) = \left\{ \frac{e_x}{d b_{xh}} + \frac{e_y}{2|y_D| Z_h} + \frac{e_x e_y}{2d|y_D| b_{xh}} \right\} \tag{4.3}$$

Considering a unit display area, from Equation 3.8 and Equation 4.3 we have:

$$g(e_x) = \left\{ \left(\frac{1}{d b_{xh}} \right) e_x + \left(\frac{1}{2R|y_D| Z_h} \right) \frac{1}{e_x} + \frac{1}{2d|y_D| R b_{xh}} \right\} \tag{4.4}$$

Calculating the first derivative of $g(e_x)$ in Equation 4.4 with respect to e_x and equat-

ing it to zero we have:

$$g'(e_x) = \left\{ \frac{1}{db_{xh}} - \frac{1}{e_x^2} \left(\frac{1}{2R|y_D|Z_h} \right) \right\} = 0 \tag{4.5}$$

By solving Equation 4.5 in terms of e_x the following result is obtained.

Result 5. *The optimal display discretization in terms of the relative error in estimating Y_h for a single 3D-point is given by:*

$$e_x = \frac{1}{\sqrt{R}} \sqrt{\frac{db_{xh}}{2|y_D|Z_h}} \quad e_y = \frac{1}{\sqrt{R}} \sqrt{\frac{2|y_D|Z_h}{db_{xh}}} \tag{4.6}$$

Proof. Follows from the same argument as Result 1 and is skipped here. □

Result 5 can be extended to optimize error over a viewing volume formed by a range of depth values [Z_{min}, Z_{max}]. Using Equation 4.5 as the basis, a MSE error metric can be defined as:

$$E(Z_h, y_D) = \left(\frac{db_{xh}}{Z_h} - 2|y_D|e_x^2 R \right)^2 \tag{4.7}$$

and we then need to optimize the following function with respect to e_x :

$$G(e_x) = \int_{Z_{min}}^{Z_{max}} \int_{-y_{max}}^{+y_{max}} \int_{X_0 - \frac{\Delta X}{2} \frac{(Z_h - Z_0)}{Z_{max} - Z_0}}^{X_0 + \frac{\Delta X}{2} \frac{(Z_h - Z_0)}{Z_{max} - Z_0}} E(Z_h, y_D)\, dX_h\, dy_D\, dZ_h \tag{4.8}$$

where

$$\begin{aligned} & y_{max} = \max(y_D), x_{max} = \max(x_{rD}) \\ & X_0 = \frac{b_{xh}}{2}, Z_0 = \frac{X_0 d}{x_{max}} \\ & \Delta X = 2 \left(X_0 - Z_{max} \frac{x_{max}}{d} \right) \end{aligned}$$

Again, calculating the derivative of $G(e_x)$ with respect to e_x and equating it to zero gives the following result [14]:

Result 6. *The optimal display discretization with respect to the average relative error in the estimation of Y_h over a viewing volume defined by a depth range is given by:*

$$e_x = \left(-\frac{db_{xh}}{2R} \frac{Z_0 I_1[Z_{min}, Z_{max}] - I_2[Z_{min}, Z_{max}]}{I_3[Z_{min}, Z_{max}] - Z_0 I_4[Z_{min}, Z_{max}]} \right)^{\frac{1}{2}} \tag{4.9}$$

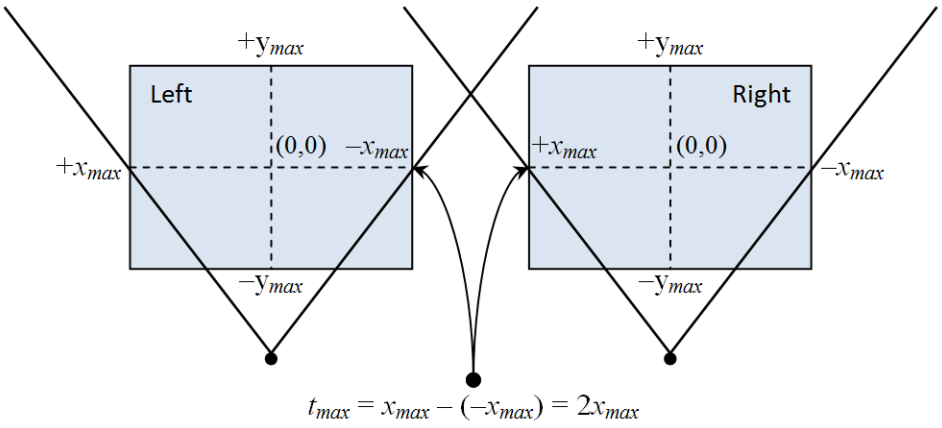


Figure 4.1: Maximum possible disparity in a typical stereo system.

where I_1 to I_4 are the integrals defined in Result 3.

Proof. Proof is similar to that of direct reconstruction from stereo images (Result 3) and is skipped here. □

4.2 General Optimization Model Over the Effective Capturing/Viewing Range

Chapter 3 and Section 4.1 consider a region in 3D defined by a set of constraints on the range of values in depth, height and the stereo Field-Of-View (FOV) and the integral of an error metric over the entire specified region is calculated and optimized to obtain the optimal discretization. The solution that is obtained in this way depends on the stereo-configuration parameters and cannot be easily applied as a general/standard solution in the design and manufacture of 3D capturing and display devices. In this section, we turn the problem into an optimization within the 2D image (display) space. The resulting optimal PAR, as follows, is related mainly to the image plane/display screen aspect ratio, whereas the effective 3D capturing/viewing volume of the stereo device is also implicitly taken into account. As a result, the solution can equally be applied to both the capturing and viewing ends of the 3D pipeline.

Recall the error metric used in Result 3, *i.e.* Equation 3.30, which is repeated here for ease of reference:

$$E_p = \left(\frac{f b_x}{Z} - |y| 2 e_x^2 R \right)^2 \tag{4.10}$$

Now, using Equation 2.2 the above error metric can be rewritten as:

$$\begin{aligned} E_p &= \left(\frac{fb_x}{\left(\frac{fb_x}{x_r - x_l}\right)} - |y| 2e_x^2 R \right)^2 \\ &= \left(x_r - x_l - |y| 2e_x^2 R \right)^2 \\ &= \left(t - |y| 2e_x^2 R \right)^2 \end{aligned} \tag{4.11}$$

where $t = x_r - x_l$ is the disparity between the corresponding points (x_r, y) and (x_l, y). Equation 4.11 can be used as the basis to transfer the optimization problem to the image plane space. Considering the discrete nature of the image plane, and assuming that x_{max} and y_{max} represent the maximum values for x and y, and t_{min} and t_{max} bound the range of expected values for disparity t (see Figure 4.1, also note that $0 \leq t_{min} \leq t_{max} \leq 2x_{max}$), the summation of error over all possible corresponding pixels in the left and right image planes can be expressed as:

$$G(e_x) = \sum_{i=m_1+1}^{m_2} \sum_{j=-n}^{n} (k - i) \left(t_i - |y_j| 2e_x^2 R \right)^2 \tag{4.12}$$

where $m_1 = \frac{t_{min}}{e_x}$, $m_2 = \frac{t_{max}}{e_x}$, $n = \frac{y_{max}}{e_y}$, and $k = \frac{2x_{max}}{e_x}$.
Considering that $t_i = ie_x$ and $y_j = je_y$, and applying Equation 3.8, $G(e_x)$ is further simplified as:

$$\begin{aligned} G(e_x) &= \sum_{i=m_1+1}^{m_2} \sum_{j=-n}^{n} (k - i) \left(ie_x - |je_y| 2e_x^2 R \right)^2 \\ &= 2 \sum_{i=m_1+1}^{m_2} \sum_{j=0}^{n} (k - i) \left(ie_x - 2je_y e_x^2 R \right)^2 \\ &= 2e_x^2 \sum_{i=m_1+1}^{m_2} \sum_{j=0}^{n} (k - i) (i - 2j)^2 \end{aligned} \tag{4.13}$$

If we define

$$H(e_x, m) = 2e_x^2 \sum_{i=0}^{m} \sum_{j=0}^{n} (k - i) (i - 2j)^2 \tag{4.14}$$

then Equation 4.13 can be written as:

$$G(e_x) = H(e_x, m_2) - H(e_x, m_1) \tag{4.15}$$

Now, expanding $H(e_x, m)$ we have:

$$H(e_x, m) = 2e_x^2(kS_1 - S_2) \tag{4.16}$$

where

$$\begin{aligned} S_1 &= \sum_{i=0}^{m}\sum_{j=0}^{n}\left(i^2 + 4j^2 - 4ij\right) \\ S_2 &= \sum_{i=0}^{m}\sum_{j=0}^{n}\left(i^3 + 4ij^2 - 4i^2j\right). \end{aligned} \tag{4.17}$$

Using the following summation formula

$$\begin{aligned} \sum_{i=0}^{n} i &= \frac{n(n+1)}{2} \\ \sum_{i=0}^{n} i^2 &= \frac{n(n+1)(2n+1)}{6} \\ \sum_{i=0}^{n} i^3 &= \frac{n^2(n+1)^2}{4} \end{aligned}$$

S_1 and S_2 in Equation 4.16 can be calculated as:

$$\begin{aligned} S_1 = \frac{mn}{6}\,&((m+1)(2m+1) + 4(n+1)(2n+1) \\ &-6(m+1)(n+1)) \\ S_2 = \frac{nm(m+1)}{12}\,&(3m(m+1) + 4(n+1)(2n+1) \\ &-4(n+1)(2m+1)) \end{aligned} \tag{4.18}$$

Considering that $e_y \ll y_{max}$, n will be large enough to assume $(n+1) \approx n$ and $(2n+1) \approx 2n$. Similarly, assuming that $e_x \ll t$, $(m+1) \approx m$ and $(2m+1) \approx 2m$. Applying these approximations, S_1 and S_2 can be approximated as:

$$\begin{aligned} S_1 &\approx \frac{mn}{3}\left(m^2 + 4n^2 - 3mn\right) \\ S_2 &\approx \frac{nm^2}{12}\left(3m^2 + 8n^2 - 8nm\right) \end{aligned} \tag{4.19}$$

From these simplified equations $H(e_x, m)$ is obtained as:

$$\begin{aligned} H(e_x, m) &= 2e_x^2(kS_1 - S_2) \\ &= -\frac{e_x^2}{6}mn\left(3m^3 - 8m^2n - 4km^2 + 8mn^2\right. \\ &\left. +12kmn - 16kn^2\right) \end{aligned} \tag{4.20}$$

Substituting m, n, and k with their corresponding definitions $m = \frac{t}{e_x}$, $n = \frac{y_{max}}{e_y}$, and $k = \frac{2x_{max}}{e_x}$ in the above equation, and after some simplification and factorization, we obtain:

$$\begin{aligned} H(e_x, t) \cong \frac{ty_{max}R}{6e_x}\left[8R^2y_{max}^2(4x_{max} - t)e_x^4\right. \\ \left. - 8Ry_{max}t(3x_{max} - t)e_x^2 + t^2(8x_{max} - 3t)\right] \end{aligned} \tag{4.21}$$

From Equations 4.15 and 4.21, and recalling that $m_1 = t_{max}/e_x$ and $m_2 = t_{min}/e_x$, we have:

$$G(e_x) = H(e_x, t_{max}) - H(e_x, t_{min}) \tag{4.22}$$

Finally, calculating the derivative of $G(e_x)$ with respect to e_x and equating it to zero leads us to the following result.

Result 7. *The optimal discretization to improve 3D estimation from a pair of stereo images is given by:*

$$e_x = \left(\frac{b + \sqrt{b^2 - ac}}{aR}\right)^{0.5} \tag{4.23}$$

where

$$\begin{aligned} a &= 24y_{max}^2[t_{max}(4x_{max} - t_{max}) - t_{min}(4x_{max} - t_{min})] \\ b &= 4y_{max}[t_{max}^2(3x_{max} - t_{max}) - t_{min}^2(3x_{max} - t_{min})] \\ c &= -[t_{max}^3(8x_{max} - 3t_{max}) - t_{min}^3(8x_{max} - 3t_{min})] \end{aligned}$$

Proof. Differentiating Equation 4.22 with respect to e_x we have:

$$G'(e_x) = aR^2e_x^4 - 2bRe_x^2 + c \tag{4.24}$$

Solving Equation 4.24 for e_x^2 while taking into account that e_x is positive, we have:

$$e_x^2 = \frac{b + \sqrt{b^2 - ac}}{aR} \tag{4.25}$$

The square root of the above equation gives the value for e_x. We can easily show that $G(e_x)$ takes its minimum at this point by performing the second derivative test for $G(x)$. □

From Result 7 and Equation 3.8, the optimal pixel aspect ratio for stereo-based 3D reconstruction is obtained as:

$$\frac{e_x}{e_y} = \frac{b + \sqrt{b^2 - ac}}{a} \tag{4.26}$$

Considering the discussion in Section 2.2, Result 7 can also be applied to the 3D estimation through a stereoscopic 3D display. Therefore, Equation 4.26 can be considered as a general optimal solution to both stereo-based 3D reconstruction and 3D perception applications. Moreover, this equation does not depend on the stereo configuration parameters such as focal length of the cameras, stereo baseline and so forth. In fact, it gives the optimal pixel aspect ratio as a function of the relative size of image plane (display) dimensions and the range of plausible disparities.

For the special case of $t_{min} = 0$, Equation 4.26 is simplified as:

$$\frac{e_x}{e_y} = \frac{t_{max}(\sqrt{2}S + 6x_{max} - 2t_{max})}{12y_{max}(4x_{max} - t_{max})} \tag{4.27}$$

where $S = \sqrt{114x_{max}^2 - 72t_{max}x_{max} + 11t_{max}^2}$.

If additionally we assume that the maximum disparity is plausible for the system, *i.e.* if we assume that $t_{max} = 2x_{max}$, then Equation 4.27 is further simplified as:

$$\frac{e_x}{e_y} = \frac{\left(\sqrt{7} + 1\right)}{6} \times \frac{x_{max}}{y_{max}} \tag{4.28}$$

where $\frac{x_{max}}{y_{max}}$ is the aspect ratio of the display screen (camera image plane). It is worth mentioning that, in practice, and especially for wider screens, the amount of disparity is bounded to a fraction of display screen width. As a result, Equation 4.26 better describes practical capturing/viewing conditions.

Chapter 5

Experimental Results and Subjective User Studies

In this chapter we present some experimental results to support the theoretical findings and derivations presented in Chapter 3 and Chapter 4. In this regard, we calculate optimal pixel aspect ratios for different stereo parameters and configurations and see how optimal PAR varies depending on different parameter settings. Moreover, based on some pixel grouping techniques, we present methods for simulating 3D viewing with different pixel aspect ratios on conventional 2D displays. These methods enabled us to conduct human observer evaluations on both simple and complex 3D objects. Our evaluation results show that given a constant total resolution it is possible to improve the 3D visual experience by choosing a finer horizontal discretization relative to the vertical discretization. Based on theoretical findings and these subjective studies, and following the usual, more practical viewing conditions, we suggest a ratio of 2:3 for pixels of a 3D capture or 3D display device.

5.1 Numerical Results

In this section we examine the optimal solutions derived in previous chapters versus some typical values for different parameters involved in these equations to show their numerical validity and soundness. In this regard, we first examine optimal models developed for a standalone stereo configuration (Results 3 and 4) by considering some parameter values that closely mimic the human visual system parameters. We then present some numerical calculations for the model developed for

stereo 3D viewing (Result 6) and also the general model that is developed for stereo reconstruction and viewing (Result 7). Again we try to use parameter values that simulate more practical capturing or viewing conditions.

5.1.1 Numerical Results for a Standalone Stereo Configuration

Table 5.1 shows a typical set of parameter values used for calculating optimal pixel aspect ratios (e_x/e_y) for with and without vergence stereo setups. Values of x_{max}, y_{max}, and R are set to 3.15 mm, 2 mm and 40635 pixels per mm^2, respectively. These values are obtained assuming that a CCD of size 6.3x4.0 mm with a resolution of 1280x800 (around 1 mega pixel) is used in the capturing process. Focal length and stereo baseline are set to 17 and 65 mm respectively, values which are close to average for the parameters of the human visual system. Table 5.1 also includes the range of values used for Z_{min} and Z_{max} in our calculations. Figure 5.1 shows the results of our calculations for parameter values in Table 5.1 based on Result 3 and Result 4 and other related equations. Figure 5.1a shows how the optimal pixel aspect ratio changes with different viewing volumes at different distances in the parallel configuration. Figure 5.1b presents corresponding results in the vergence configuration using three different small vergence angles, α = 1, 3, and 5 degrees, over the same depth ranges and viewing volumes. The corner of the original graph is magnified in order to highlight the effect of applying different vergence angles. It can be seen that for small vergence angles there is no significant difference either between the top and bottom graphs or between results obtained for different small vergence angles on the bottom graph. Based on this observation, we can say that at least for small vergence angles the effect of vergence in determining the optimal pixel aspect ratio is negligible. It should be noted that for a stereo baseline of 65 mm, a 5 degree vergence angle is equivalent to focusing on a point about 371 mm from cameras/viewing point, which is a good representation of a practical situation.

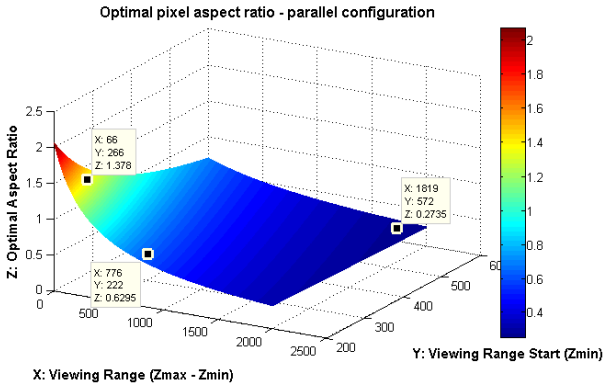
Another observation from these graphs is how the optimal pixel aspect ratio changes relative to the viewing range start point (Z_{min}) and length of the viewing range ($Z_{max} - Z_{min}$). For closer start points with smaller viewing ranges, a larger optimal ratio is obtained; but if the start point is far enough or the viewing range is large enough, a smaller aspect ratio is calculated. Moreover, these graphs reveal that even though the optimal points are obtained by optimization of the relative error in estimating the Y component, the optimal aspect ratio is smaller than one in

Table 5.1: Parameter values used for calculating the optimal pixel aspect ratio for a stereo camera.

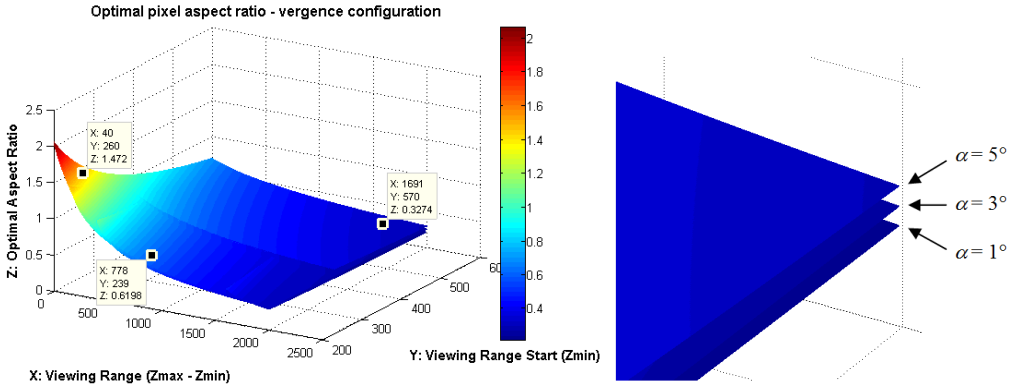
Parameter	**Value**(mm)	**Parameter**	**Value**
f	17	Z_{min}	From 200 mm to 600 mm
b_x	65	$Z_{range}(Z_{max} - Z_{min})$	From 1 mm to 2000 mm
x_{max}	3.15	Resolution (R)	1280×800 (40635 p/mm^2)
y_{max}	2		

Table 5.2: Some optimal pixel aspect ratios calculated for both without- and with-vergence configurations using parameter values mentioned in Table 5.1.

	Z_{min}(mm)	Z_{max}(mm)	α(Degree)	Z_0(mm)	Z_{int}(mm)	K	e_x/e_y
1	200	1000	0	175.3968	-	-	0.6315
2	180	650	0	175.3968	-	-	0.9007
3	500	2500	0	175.3968	-	-	0.2687
4	200	1000	3	135.3982	599.4954	1.0097	0.5844
5	200	940	5	117.2110	359.0514	1.0162	0.6153
6	180	650	3	135.3982	599.4954	1.0097	0.9538
7	180	650	5	117.2110	359.0514	1.0162	0.8747
8	500	2000	3	135.3982	599.4954	1.0097	0.3143
9	500	2000	5	117.2110	359.0514	1.0162	0.3718



(a) Parallel camera configuration

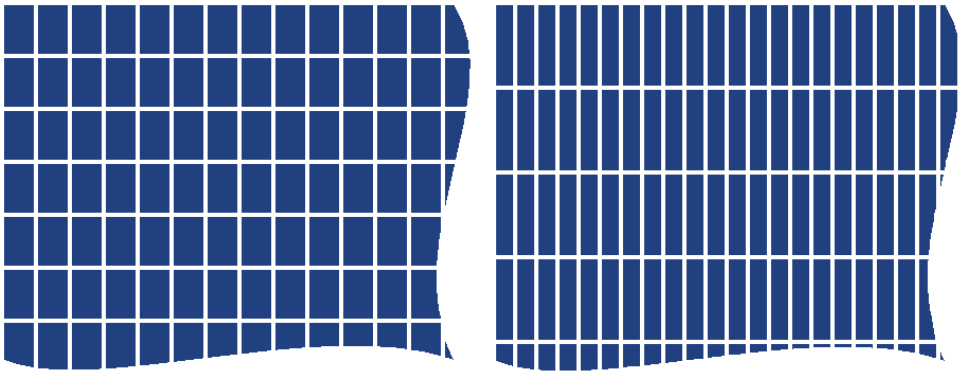


(b) Vergence camera configuration (vergence angles α = 1, 3, and 5 degrees)

Figure 5.1: Variation of optimal pixel aspect ratio for parameter values listed in Table 5.1

most configurations (and in fact in more practical configurations). This means that the X and Z estimates are improved as well, compared to the conventional pixel arrangement where pixels are equally distributed over the x and y axes.

Table 5.2 presents a set of optimal aspect ratios and some other parameters calculated for a number of typical start points and viewing ranges. As noted above, the optimal ratio may vary significantly depending on viewing distance and volume. For example, if we optimize for an object appearing between 200 and 1000 mm, *i.e.*, within one meter of the camera (Table 5.2 - row 1), then the optimal pixel aspect ratio will be calculated as 0.6315, which after rounding off is approximately equal to a non-square discretization of 2:3 (pixel-width:pixel-height) aspect ratio or equivalently filling the image plane (CCD) with vertically rectangular pixels of height 1.5 times longer than their width. If an object appears between 500 and 2500 mm (Table 5.2 - row 3) the optimal pixel aspect ratio will be 0.2687, which is close to a 1:4 pixel discretization. Figure 5.2 shows these two non-square pixel



(a) (b)

Figure 5.2: Filling the display screen with non-square pixels: (a) 2:3 discretization *i.e.* three horizontal pixels *vs.* two vertical pixels, and (b) 1:4 discretization.

distributions.

5.1.2 Numerical Results for 3D Displays

Table 5.3 shows a typical set of values used for calculating optimal PARs (e_x/e_y) for stereoscopic 3D displays. Here, values of x_{max}, y_{max}, and R are obtained assuming that a 14.1" display with 1280x800 resolution is used. As in the case of a standalone stereo camera (Table 5.1), the focal length f_h and baseline b_{xh} are selected to be close to the human vision system parameters.

Figure 5.3 shows the computational results for values mentioned in Table 5.3 based on Result 6 formulations. As in the case of standalone stereo camera configurations (compare this figure with Figure 5.1), for closer minimum depth (Z_{min}) with smaller ranges $(Z_{max} - Z_{min})$, a larger ratio is obtained, but if the minimum depth is far enough or the viewing-range is large enough, a smaller optimal pixel aspect ratio is calculated. Specifically, for the range 150-600 mm, which is a practical range for this configuration, the optimal ratio is calculated as 0.6621 or approximately 2:3 (3 horizontal *vs.* 2 vertical pixels). Regarding the discussion in Section 2.2 these results can be applied irrespective of the stereo configuration in the capturing side (see also Section 5.3).

Table 5.3: Values used for calculating optimal pixel aspect ratio for a stereoscopic 3D display.

Parameter	**Value** (mm)	**Parameter**	**Value**
f_h	17	R	17.76 pixel/mm^2
b_{xh}	65	d	500 mm
x_{max}	303.7021	Z_{min}	From 200 to 400 mm
y_{max}	189.8138	$Z_{range}(Z_{max} - Z_{min})$	From 1 to 800 mm

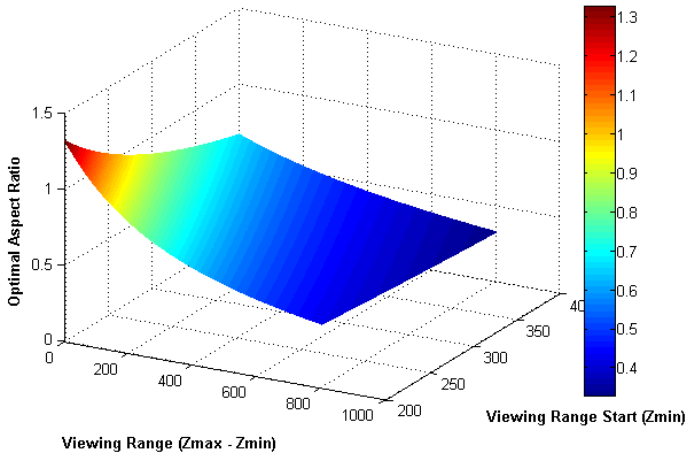
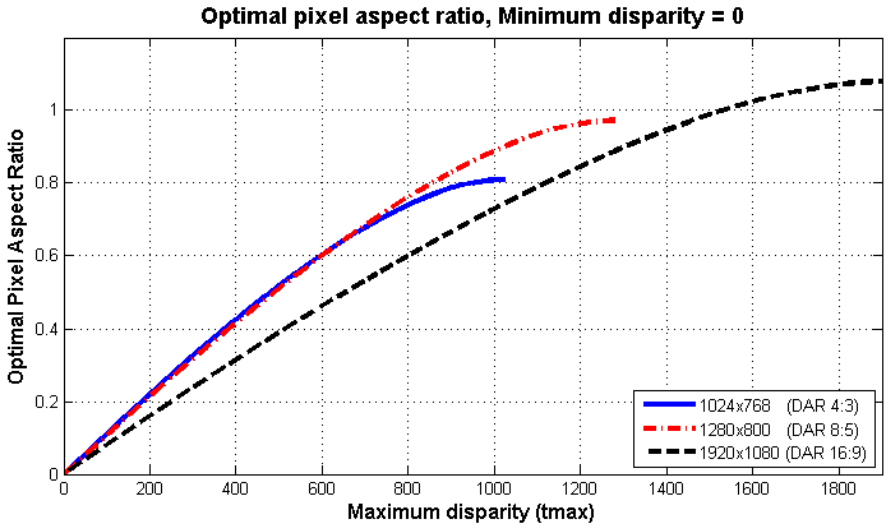


Figure 5.3: Optimal pixel aspect ratio changes for values mentioned in Table 5.3.

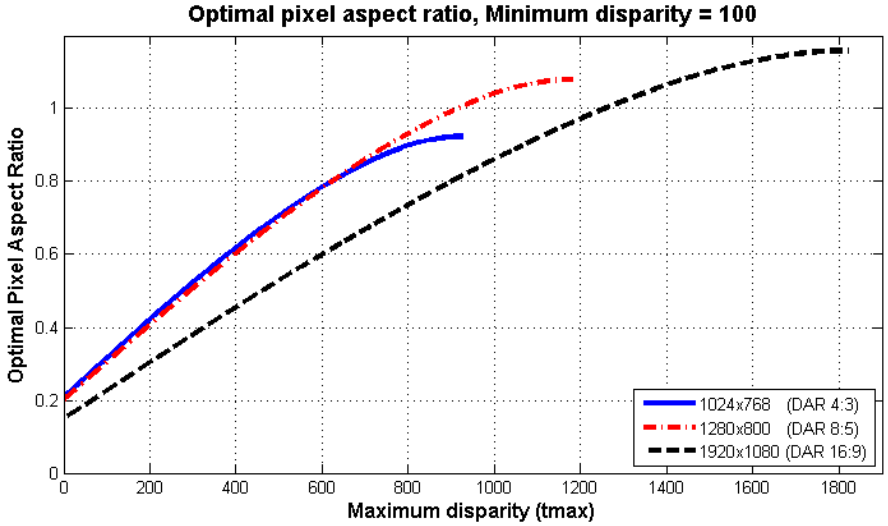
5.1.3 Calculating a Unified Optimal PAR for Stereo Capturing and 3D Viewing

The optimal PARs that are calculated using Results 3, 4 and 6 depend on stereo configuration parameters such as the stereo base-line and the focal length of the cameras, or viewing conditions such as the viewing distance or size of the 3D display. In practice, we may follow more practical capture and viewing conditions to calculate an appropriate PAR. Alternatively, we can use Result 7 to calculate an optimal PAR that can universally be applied throughout the 3D pipeline, as follows.

Figure 5.4 shows the optimal PARs calculated using Result 7. This figure shows changes of optimal pixel aspect ratio as a function of maximum plausible disparity t_{max} for displays (imaging sensors) of different aspect ratios, namely 4:3, 15:9, and 16:9, and for two cases where minimum disparity t_{min} is set to zero and 100, respectively. Figure 5.4a is representative of scenes with some objects/background at infinity and Figure 5.4b is representative of some indoor scenes where even the



(a)



(b)

Figure 5.4: Optimal pixel aspect ratio as a function of maximum plausible disparity, t_{max}, for displays (imaging sensors) of different aspect ratios: (a) Minimum disparity, t_{min}, is set to zero, and (b) Minimum disparity is set to 100.

background objects might have some disparities.

As can be seen in these figures, the optimal PAR becomes larger with wider disparity ranges, *i.e.*, becomes larger with having larger capturing or viewing volumes. The reader may have noticed that this is opposite to the behavior of the other optimal solutions depicted in Figures 5.1 and 5.3. The explanation lies in the way that the optimal solution is calculated in Results 3, 4, and 6. The integral in these optimizations overweighs the voxels of bigger size, and as the size of voxels in-

creases with the distance to the viewer or camera (see Figure 2.9), these models suggest sharper (smaller) optimal PARs with larger capturing or viewing volumes. In contrast, the optimization process in Result 7 uses a summation over a discrete space and equally treats all voxels within the optimization range.

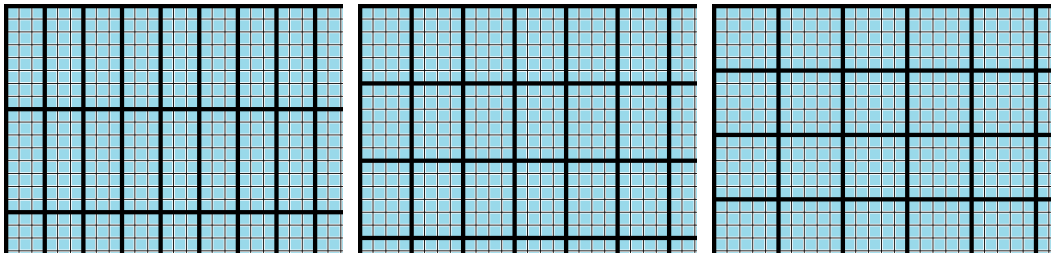
In general, the optimal PAR suggested by Result 7 remains smaller than 1 except for those extreme cases of having very large disparities which are practically impossible. For more practically plausible disparity ranges, in which, depending on the size of the display, we may assume that a maximum disparity t_{max} of up to $1/4^{th}$ to $1/3^{rd}$ of the display width is plausible, the optimal pixel aspect ratio changes from 0.35 to 0.65. For example, for a display of 8:5 ($\frac{x_{max}}{y_{max}} = \frac{8}{5}$) aspect ratio, assuming the disparity is bounded within 10 to 35 percent of the screen width ($t_{min} = 0.10 * 2 * x_{max}$ and $t_{max} = 0.35 * 2 * x_{max}$), from Equation 4.26 the optimal pixel aspect ratio is calculated as 0.58. Similarly, for a display with aspect ratio of 16:9, if we assume that $t_{min} = 0$ and the maximum possible disparity is bounded to 35 percent of the display width ($t_{max} = 0.35 * 2 * x_{max}$) then using Equation 4.27 the optimal pixel aspect ratio is calculated as 0.51. For the spacial case of having a unit display or imaging area ($\frac{x_{max}}{y_{max}} = 1$), from Equation 4.28, the optimal pixel aspect ratio is calculated as 0.61. In other words, if we optimize for a unit display or imaging area and take into account all possible disparities, *i.e.* $t_{min} = 0$ and $t_{max} = 2 * x_{max}$, then an optimal pixel aspect ratio of 0.61 or approximately 2:3 (3 horizontal *vs.* 2 vertical pixels) is obtained again, irrespective of the stereo imaging or stereo display configuration parameters.

5.2 Subjective User Tests with Stereo 3D Models

We have conducted a set of user tests to validate our theoretical results with subjective user evaluations. Our studies are based on simulating different pixel aspect ratios on a conventional display. In practice, the standard ratio may be determined based on user studies on actual display prototypes with vertically rectangular pixels of different pixel aspect ratios.

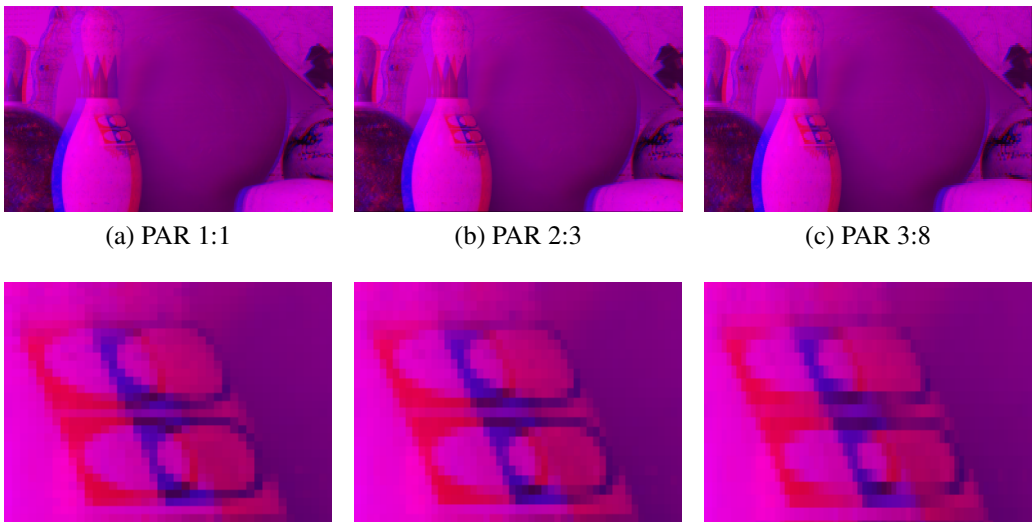
5.2.1 Simulating Different PARs on Capturing and Viewing Sides

Since we were using conventional 2D displays in our subjective studies and it was impossible to adjust the actual pixel aspect ratio on these screens, we needed to devise some techniques to simulate different discretization ratios along the horizontal



(a) (b) (c)

Figure 5.5: Simulating different pixel aspect ratios by grouping neighboring pixels as a single pixel while preserving the same total resolution by devoting almost the same number of pixels to each virtual pixel in all three configurations: (a) 3:8, (b) 2:3, and (c) 1:1 pixel aspect ratio.



(a) PAR 1:1 (b) PAR 2:3 (c) PAR 3:8

(d) PAR 1:1 (e) PAR 2:3 (f) PAR 3:8

Figure 5.6: Conventional uniform pixel distribution *vs.* horizontally finer pixel distributions for the same total resolution $R \cong 42000$. First row: (a) PAR (Pixel Aspect Ratio) = 1.0, (b) PAR = 0.66 , and (c) PAR = 0.38. Second row: Enlarged images corresponding to the part of the pin in first row images to see the differences in these pixel arrangements.

and vertical axes. For this purpose, we established a virtual stereo imaging system with arbitrarily sized rectangular pixels. This virtual stereo imaging system accepts 3D structure descriptions such as 3D triangular meshes or 3D point clouds as input and generates stereo projections based on the given stereo configuration.

On the display side, we used a group of neighbouring pixels of a conventional display to simulate different pixel aspect ratios. For example, by grouping a set of 5x5, 6x4, or 8x3 neighbouring pixels together as a single pixel we can simulate 1:1,

2:3, and 3:8 pixel aspect ratios, respectively, while the total resolution is kept almost the same for these three combinations. Figure 5.5 shows a pictorial description of this process. We have also used this technique to create test sets from real stereo images. We used pure red and blue colors for generating the left and right views, respectively. A pair of simple anaglyph red-blue glasses was used to guide the right and left views to the corresponding eyes of the viewers. Figure 5.6 shows samples of stereo images simulated in this way. The figure shows how different pixel distributions look like for two rational PARs, *i.e.*, 2:3 and 3:8 (Figures 5.6b and 5.6c, respectively), compared to the conventional pixel distribution with 1:1 PAR (Figure 5.6a) for the same total resolution. These images must be viewed using Red-Blue anaglyph glasses and preferably in their original size for the best 3D effect. However, to better demonstrate the differences between these pixel arrangements, part of the pin in the images of the first row is enlarged and represented in the second row. In general, even though this method is a bit restrictive in terms of selecting any arbitrary pixel aspect ratio, it is a very simple, inexpensive technique for achieving different pixel aspect ratios on conventional displays in the absence of actual (prototype) display devices having none-square pixel arrangements.

5.2.2 Subjective Evaluation Criteria

The evaluation criteria have been chosen from the 3D visual experience model proposed in [94]. These criteria include *sense of depth*, *picture quality* and *overall sense*. Sense of depth refers to the extent to which objects are distinguishable and well-shaped across the depth dimension. Picture quality refers to the quality of the perceived image, which may be influenced by several factors such as blockiness, brightness, noise and blurriness. Here, the picture quality is affected mainly by the non-similar discretization across the vertical and horizontal dimensions. Overall sense measures viewer's overall satisfaction with the given stereo 3D representations. Once again, these representations differ only in the aspect ratio of the pixels, while the total resolution is kept the same.

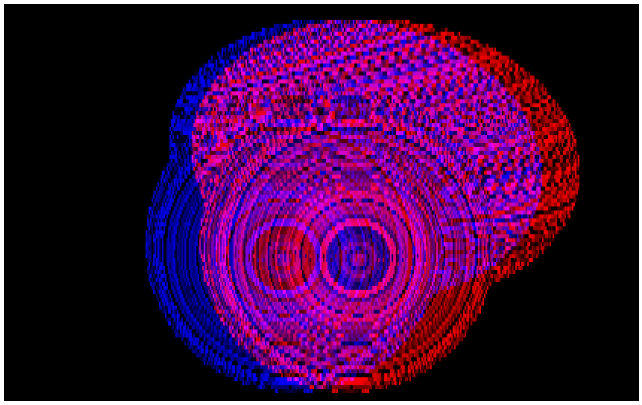
5.2.3 Statistical Analysis of Subjective User Tests

In our subjective studies, depending on the experiment setup, two or more samples of user preferences or ranks given to the various PAR representations of the same image(s) are examined for the null hypothesis, *i.e.*, equality of the mean of drawn samples (see Sections 5.2.4 and 5.2.5). Depending on the number of groups

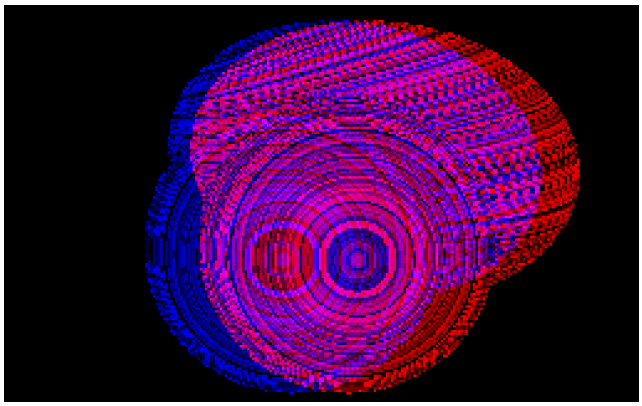
(number of PARs) involved in the experiment, either the T-Test or the analysis of variance (ANOVA) is used to examine the statistical significance among groups. The T-Test is used if only two PARs are involved in the experiment or two specific PARs are needed to be compared among several PAR representations. ANOVA is used if more than two PARs are involved in the experiment. The result of a T-Test is usually reported in the form of a *P*-value that shows how certain we can be that the null hypothesis is false. On the other hand, the results of an ANOVA test are often reported in the form of a table (see Table 5.4). The table shows and compares two sources of variation, *i.e.*, between-groups and within-groups variations. Between-groups variation is the sum of the squares (*SS*) of the differences between group means and the average of all the sampled items (grand mean). Within-groups variation is measured by summing the square (*SS*) of the distances of each value to the corresponding sample mean. These values are divided by the corresponding degree of freedom (*df*) to obtain the mean square (*MS*) of the variation. The test statistic *F* is calculated as the ratio of the in between and within groups *MS*, and in combination with the two degrees of freedom, it is used to find the *P*-value from the *F* distribution. The larger the F or equivalently the smaller the P-value, the more certain we are that the null hypothesis is false [78].

5.2.4 Experiment 1 - User Tests with Synthetic Stereo Images

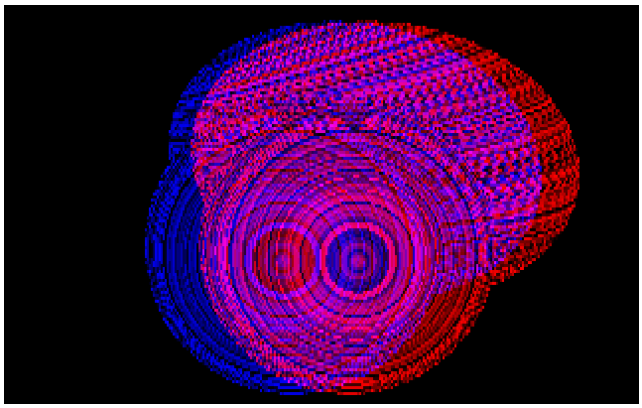
Experiment 1 Setup: Figure 5.7 shows a set of three red-blue images used to conduct our first set of user tests. These images are generated from a synthetic 3D model with three different PARs (3:8 (8x3), 2:3 (6x4), and 1:1 (5x5)) using our virtual stereo imaging system. Here, the model is composed of a set of 3D points which collectively describe a sphere with a radius of 19 cm and an ellipsoid with semi-minors of 50, 15, and 35 cm. The centers of the sphere and ellipsoid are located at a distance of 40 and 80 centimeters, respectively, from the stereo camera. Two arbitrary rotations are also applied to the ellipsoid. The colors (intensities) are randomly assigned to the 3D scene points, but geometric shape of the objects is preserved by assigning the same color to the points located on the same orbit. For this test set, the imaging and displaying systems are configured using the values shown in Table 5.1 and Table 5.3. Considering the depth range filled by these two objects, the theoretical optimal PAR is given by the first row in Table 5.2, *i.e.* 0.6315, which is almost equivalent to an aspect ratio of 2:3, the ratio that is used to generate the middle image in Figure 5.7.



(a) Image 1, PAR 3:8



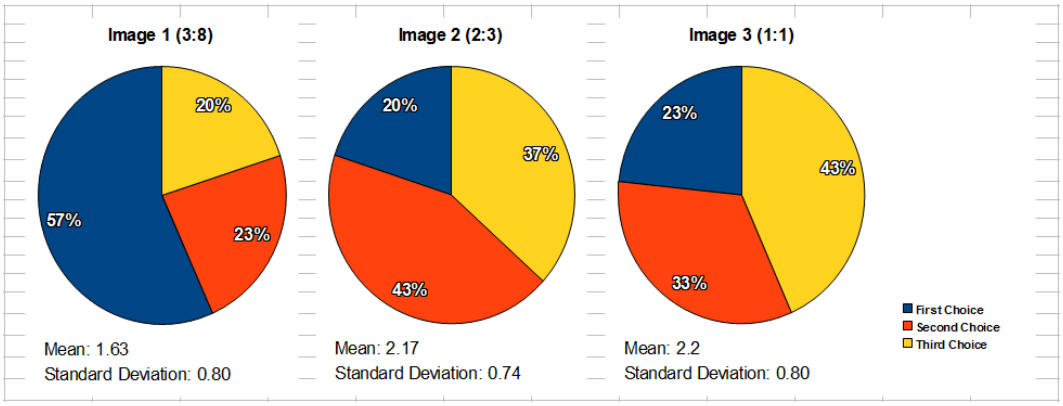
(b) Image 2, PAR 2:3



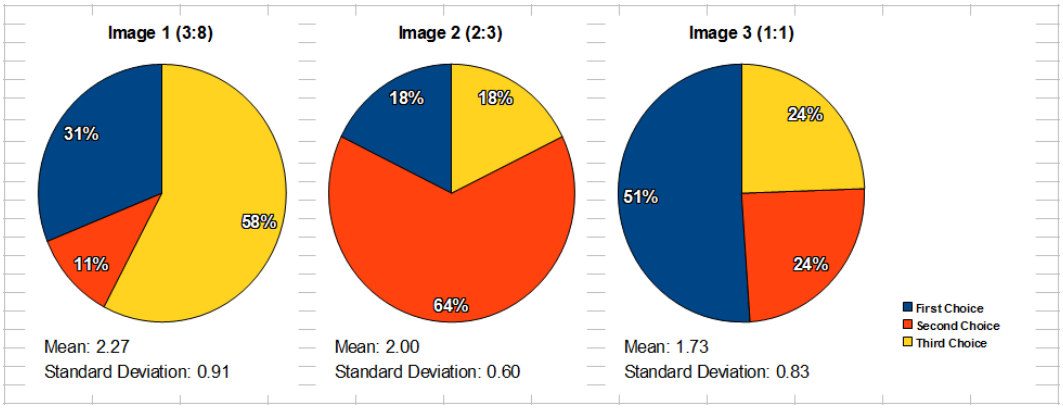
(c) Image 3, PAR 1:1

Figure 5.7: Red-blue stereo pairs generated from a synthetic scene with different PARs.

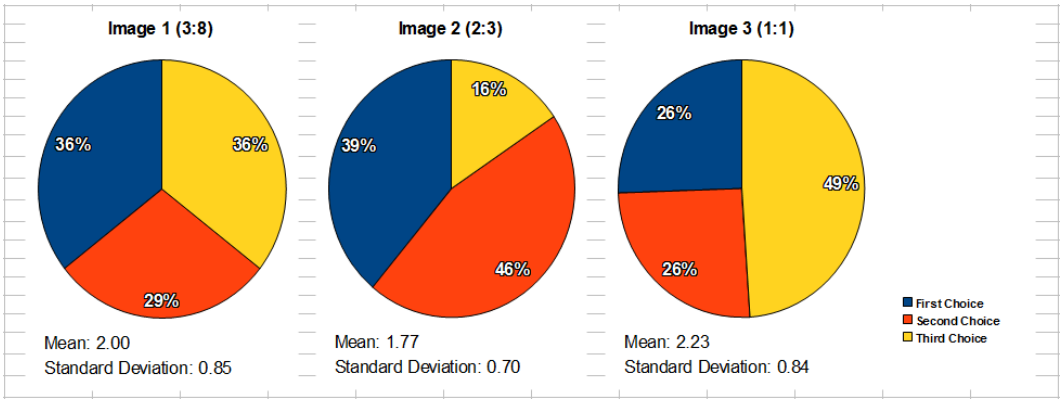
Fifteen subjects (university graduate students) participated in this experiment. Participants were asked to rank the images according to the above-mentioned criteria, *i.e.*, sense of depth, picture quality, and overall sense. Before ranking, we asked all participants to describe the reconstructed 3D scene to make sure that all of them



(a) Sense of Depth



(b) Picture Quality



(c) Overall Sense

Figure 5.8: Results of conducting subjective tests with images shown in Figure 5.7.

were able to clearly perceive the ellipsoid and the sphere in front of it. The entire screen, 14.1" display with 1280x800 resolution, was used to display an image, and the viewers were able to switch forward and backward to view all three images for comparison. We did not fix any time limit for the viewers and they could look at the 3D pictures as many times as they needed to decide on the rankings.

Experiment 1 Results: The pie charts in Figure 5.8 illustrate the result of this subjective study. The figure also presents some statistics for each evaluation criterion including mean and variance in each group (image). These results suggest an association between the sense-of-depth and pixel aspect ratio. As depicted in Figure 5.8a, 57 percent of viewers have the best sense of depth with Image 1, which has a pixel aspect ratio of 3:8, 20 percent have the best sense of depth in Image 2, with a pixel aspect ratio 2:3, and 23 percent selected Image 3, with a regular 1:1 pixel aspect ratio. In other words, a majority of the viewers preferred Image 1 (average rating 1.63), which has the finest horizontal resolution, as their first choice. This result, in general, is consistent with our theoretical results showing that images with horizontally finer resolution provide better sensation of depth.

For ranking against picture quality (Figure 5.8b), viewers were asked to rank images regardless of any perception of depth. Again, the results show a meaningful but inverse relationship between pixel aspect ratio and picture quality. More than 50 percent selected Image 3 for the best picture quality (average rating 1.73). As discussed in Section 3.1, this may be the result of image degradation that happens in the vertical direction when the discretization becomes coarser in that direction, and as a result the Y component estimation error might be increased. However, this trade-off of having finer horizontal discretization by sacrificing some quality in the Y component improves both the X and Z components, and therefore the viewers have a better overall sense preference in Image 1 and Image 2, which have non-square discretization (Figure 5.8c). We observe in Figure 5.8 that in terms of overall sense, Image 2, with aspect ratio 2:3, has been the first or second choice for most of the viewers (average rating 1.77). This result illustrates certain compromise in the viewers' decisions towards the trade-off between sense of depth and picture quality.

In general, the results of Experiment 1 can be considered as supporting evidence for our theoretical deductions. This experiment is also worthwhile in terms of showing how different quality factors may change with different PAR representations. However, it should be noted that the ANOVA test on Experiment 1 samples does not give strong evidence for rejecting the null hypothesis based on this small set of samples. The one-way ANOVA test P-Values for examining the difference in mean of the samples are 0.14, 0.23, and 0.31 for sense of depth, picture quality, and overall sense, respectively. Nevertheless, these P-values are aligned with our assumptions, and as shown in Experiment 2 we expect more statistically significant results with a growing number of samples.

(a) Aloe (b) Bowling

(c) Rocks (d) VP (e) Chinese dragon

Figure 5.9: Set of images used in Experiment 2 - user evaluation with real images and objects.

5.2.5 Experiment 2 - User Tests with Real Stereo Images

Experiment 2 Setup: We conducted a second round of user tests to study the effects of the pixel aspect ratio on 3D perception using real stereo scenes and objects. We simplified evaluation criteria in this experiment and asked the users to compare the images only in terms of their overall personal preference. In this experiment we used several scenes with various textures and structural patterns and features, shown in Figure 5.9. Figure 5.9a, Aloe, has slanted textural patterns and sharp edges or lines in all directions. Figure 5.9b, Bowling, is composed of mainly rounded objects. Figure 5.9c, Rocks, has some sharp horizontal edges with some other mainly horizontally aligned textural patterns, whereas in Figure 5.9d, VP, vertical structural edges have superiority over other visible features. Finally, 5.9e, Chinese Dragon, is an image rendered from the Chinese Dragon 3D model that contains a combination of regular and irregular line and textural patterns.

As described earlier, we used the pixel grouping technique to create images with different PARs from real stereo images. For each scene, stereo pairs with different pixel aspect ratios but the same total resolution were generated and the users were asked to choose their preferred ones in a pair-wise manner. More precisely, assum-

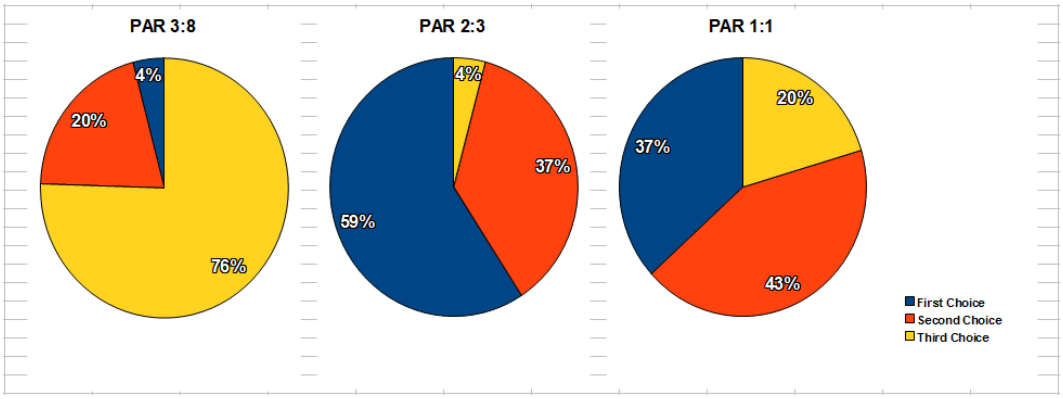


Figure 5.10: Summary of the results of the subjective tests (Overall Personal Preference) on real stereo pairs for 3:8, 2:3, and 1:1 PARs.

ing that I_1, I_2, and I_3 are the three stereo images generated for scene S with three different PARs 1:1 (5x5), 2:3 (6x4), and 3:8 (8x3), we created three pairs I_1I_2 (or I_2I_1), I_2I_3 (or I_3I_2), and I_1I_3 (or I_3I_1) of these images (the order of images was picked at random). Images (b), (c), (d), and (e) of Figure 5.9 were used for generating these test cases. We put all these pairs together and shuffled them to form the test set. We enriched the test set by randomly interleaving some other cases that were generated with some finer total resolutions to examine how users treat the images when the noise introduced by pixel grouping technique is less apparent. These finer test cases were generated from images (a), (b), (c), and (d) of Figure 5.9. These test cases were generated by grouping 4x4 and 5x3 pixels to examine 1:1 PAR versus 3:5 (2:3) PAR, and by grouping 3x3 and 4x2 pixels to compare 1:1 PAR versus 1:2 PAR.

Thirteen subjects participated in this experiment, the majority of them being university graduate students. During the test, the users were asked to show their preference toward one of the stereo images in each pair. As in Experiment 1, we did not enforce any time limit, but all subjects spent almost the same amount of time (between 25 to 30 minutes) to evaluate all cases. We also did not enforce a specific viewing distance, but we recommended the viewing distance of 70-80 cm considering that we were using wider display screen (19" display) in this experiment.

Experiment 2 Results: Figure 5.10 shows the user evaluation results for 5x5, 6x4, and 8x3 groupings (1:1, 2:3, and 3:8 PARs, respectively). As can be seen from the pie charts, the results are quite different for these three groups in favor of the images with a pixel aspect ratio 2:3 (6x4 pixel grouping). The outcome of the ANOVA test (F = 52.19, P-value = 8.99E-18) also confirms the substantial difference between

Table 5.4: Single factor ANOVA test for comparing user evaluation results on real stereo images for 3:8, 2:3, and 1:1 PAR groups.

Groups	Count	Sum	Average	Variance
PAR 3:8	49	133	2.71	0.29
PAR 2:3	49	71	1.45	0.34
PAR 1:1	49	90	1.84	0.56

ANOVA: Single Factor			
Source of Variation	SS	df	MS
Between Groups	41.18	2	20.59
Within Groups	56.82	144	0.39
F = 52.19	$F_{critical}$ = 3.06	P-value = 8.99E-18	

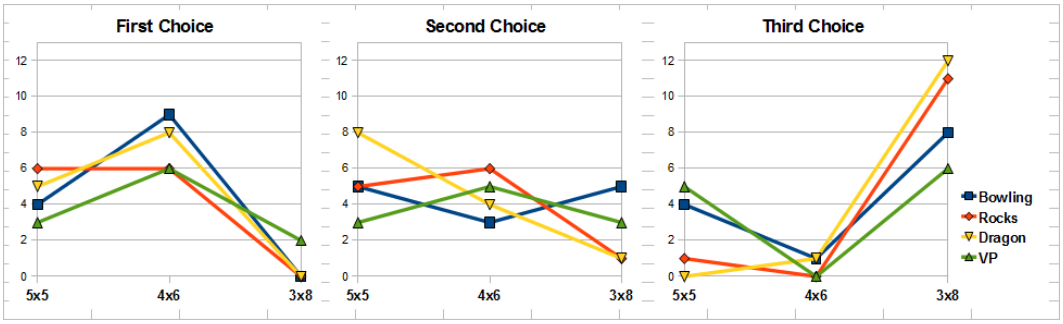


Figure 5.11: Comparing subjective test results for different scenes used in the second experiment.

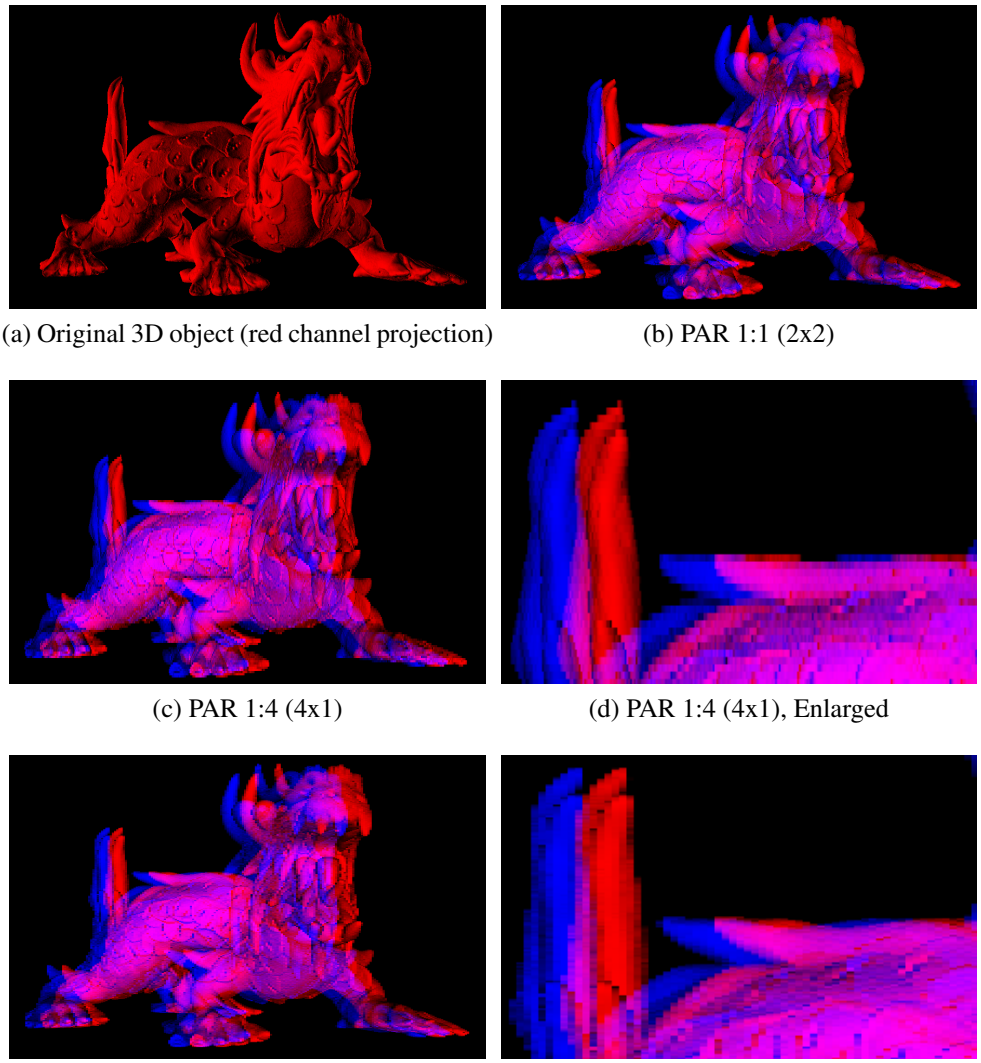
groups (see Table 5.4). PAR 2:3 not only was selected as the first choice by many viewers (59 percent) but also rarely was the third choice of viewers (4 percent). In other words, in 96 percent of the cases, images with PAR 2:3 are selected either as the first or the second preferred image. In contrast to the results of Experiment 1 (synthetic scene), here images with PAR 1:1 (i.e. images with square pixel distribution) mainly occupied second place (43 percent) and images with PAR 3:8 were often selected as the least favorite ones (76 percent). This can be explained by the way that pixel grouping technique treats image features. In fact, the real images we used in this experiment possess more textural or structural features than the simple synthetic scene of Experiment 1, which is composed of poorly featured geometrical objects. As a result, these images are more prone to the noise introduced by sharper simulated PARs.

Figure 5.11 shows a more detailed representation of the results presented in Figure 5.10 pie charts. The line charts of this figure show how different scenes are treated by viewers. In general, the drawings show a consistent behavior toward all four different scenes that are used in this experiment. However, the results seem to be affected to a certain extent by the scene structures. For example, for the PAR 1:1 representation (5x5 grouping), the users more frequently selected the Rocks scene, which has more horizontally oriented edges and textural features, as their first choice than the VP scene, which has more vertically aligned structures. However, the differences are not that significant that contradict the general trend and the expected theoretical results.

We separately analyzed the results of the test cases with finer total resolution. For 3x3 versus 4x2 test cases, 54 percent of viewers were in favor of images with square pixel distribution (PAR 1:1 (3x3)). We did not find a statistically significant difference between these two groups. However, at least, we can say that the images with PAR 1:2 (4x2), which have less apparent pixel grouping noise, have received much more attention than images with PAR 3:8 (8x3) when compared to the images of corresponding square pixel distribution. For 4x4 versus 5x3 cases we found that 63 percent of viewers were in favor of images with non-square pixel distribution (PAR 3:5 (5x3)) with a statistically significant difference between two groups (paired T-Test P-value = 0.01). These results are consistent with those obtained for the images of the same PARs with coarser total resolution (see pie charts in Figure 5.10).

5.2.6 A Closer Look at the Effects of Non-square Pixel Discretization on Picture Quality

Figure 5.12 represents more clearly the image quality degradation for a coarser discretization in the vertical or horizontal direction. These images are generated from the Chinese Dragon 3D point cloud using our virtual stereo imaging system with 1:4, 1:1, and 4:1 pixel aspect ratios, respectively, and with the same total resolution. For the PAR 1:4 image (Figure 5.12c) the discretization is more apparent along the horizontal lines. This is evident, for example, in the horizontal supplementary tail over the vertebral line of the dragon. On the other hand, for the PAR 4:1 image (Figure 5.12e) the degradation is more apparent along the vertical lines. This is perceptible, as an example, in the dragon's vertical tail. Both of these images reveal lower image quality in some parts compared to the conventional square pixel



(a) Original 3D object (red channel projection)

(b) PAR 1:1 (2x2)

(c) PAR 1:4 (4x1)

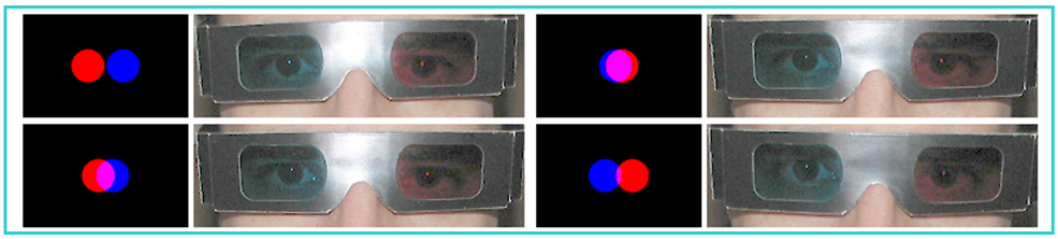
(d) PAR 1:4 (4x1), Enlarged

(e) PAR 4:1 (1x4)

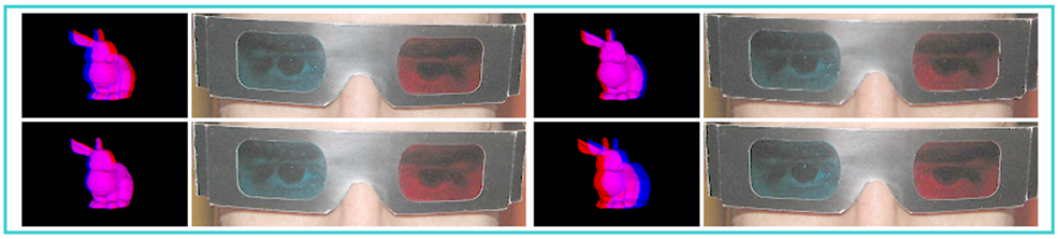
(f) PAR 4:1 (1x4), Enlarged

Figure 5.12: Samples generated from the Chinese Dragon 3D point cloud with different pixel aspect ratios.

discretization in the PAR 1:1 image (Figure 5.12b). In general, for complex scenes composed of a set of edges, line, or textural features, depending on the feature directions, non-square pixel discretization may degrade the picture quality in parts while quality may be improved in some other areas. However, since a finer resolution in the horizontal direction also improves the 3D estimation (sense of depth), and also considering that the optimal PAR reduces the maximum error in the estimation of Y, a better 3D picture is experienced overall when an appropriate (optimal) PAR is selected. This enhanced 3D experience can also be explained by the way the visual cortex processes disparity information. The primary visual cortex only responds to



(a)



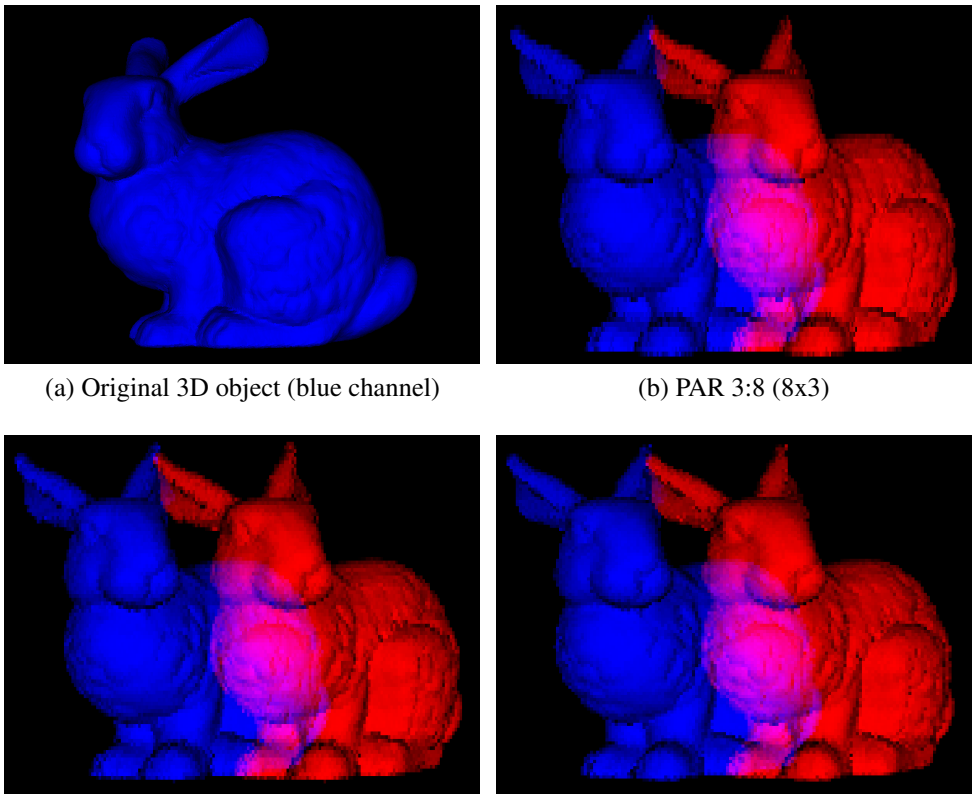
(b)

Figure 5.13: Orientation of eyes in response to (a) different disparities and (b) different stereo capturing vergences.

the absolute disparity of visual stimuli [32], *i.e.*, the difference in the location of a feature within the left and right retinal images with respect to the anatomical landmarks on the left and right retinas [12]. On the other hand, the finest stereoacuity judgments are psychophysically generated in higher processing levels and based on the stimuli that contain relative disparities, *i.e.*, the differences in the absolute disparities of different visual features [68, 102]. Our model puts more emphasis on the horizontal axis. This provides more accurate disparity cues to the primary visual cortex, which in turn may lead to more accurate judgment on relative disparities.

5.3 Effects of Vergence on 3D Viewing

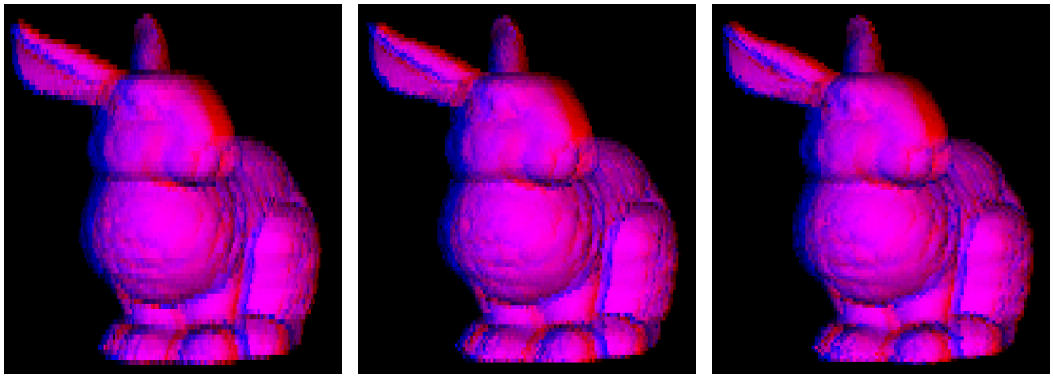
To understand the behavior of human eyes when they are watching stereo content on a 3D display, we tracked their reaction to the changes in disparities and also changes in stereo capturing vergence. Figure 5.13 shows a subset of red-blue images used in these tests. The two upper rows (Figure 5.13a) are samples of stereo images rendered with different disparities and the corresponding images of the eyes when watching these images. The images roughly show that the eyes' orientation is almost independent of the amount of disparity. The two bottom rows (Figure 5.13b) are the stereo pairs generated from the Bunny 3D mesh under different vergence angles using the virtual stereo imaging system we have established for this purpose



(a) Original 3D object (blue channel) (b) PAR 3:8 (8x3)

(c) PAR 2:3 (6x4) (d) PAR 1:1 (5x5)

Figure 5.14: Samples generated from the Bunny 3D mesh in parallel camera configuration.



(a) PAR 3:8 (8x3) (b) PAR 2:3 (6x4) (c) PAR 2:3 (1x1)

Figure 5.15: Samples generated from the Bunny 3D mesh in which vergence is included.

(see Section 5.2.1). Again, we can see that, in accordance with our assumption in Section 2.2, the eyes reveal almost the same behavior in dealing with the stereo pairs generated under different vergence configurations.

Apart from the effects of capturing vergence on the reaction of the eyes, which seems not to be significantly affected, the vergence can be sometimes beneficial in providing stereo content. In order to study these benefits we compared the effect of visualization at close range with and without vergence. Figure 5.14 shows a test set generated for the Bunny 3D mesh under parallel camera configuration. As depicted in this figure the amount of disparity for this small model is quite large. This makes it difficult for some people to easily merge the left and right views and may cause eye strain after a short period of time. Vergence can help control the amount of disparity in the stereo output. Even with a small vergence angle the amount of disparity is significantly affected. Figure 5.15 shows the images generated for the same model as in Figure 5.14, but incorporating a vergence angle of approximately 3 degrees. These images are much easier to fuse together for a human observer. However, vergence influences the 3D output in several ways. In fact, the amount of disparity affects the extent of "popping-out-of" or "sinking-into" the screen. In other words, focusing in front of the object moves it farther away and focusing at the back of the object brings it closer to the viewer of a 3D display. Although these effects may help provide enhanced impressions for some 3D content, they may also cause some undesired side-effects such as improper scale or unnatural shape [56].

Part II

Sampled Object Representation and Visualization

Chapter 6

Sample-based Object Representation

Many visual media content representations can theoretically be considered as samples of some specific, often lower dimensional cases of a generic 7-dimensional plenoptic function $I = f(x, y, z, \phi, \theta, \lambda, t)$. This function, in its general form, gives the amount of the radiance received along any direction $V(\phi, \theta)$ arriving at any point $P(x, y, z)$ in space, at any time t and over any range of wavelength λ. Such a complete representation would implicitly include a description of any possible photograph that could be taken of any particular part of the world at any time [4]. Assuming that the intensity of rays does not change along their direction, then the points representing a motionless 3D object can be thought as samples of a 6-dimensional (time-independent) plenoptic function measured at the surface of the object. Dynamic 3D objects can be represented in the same way but with an additional time argument (7-dimensional plenoptic function). Here, we may assume that the surface sampling process is repeated in appropriate time intervals to describe changes in the position of samples (and other attributes such as intensity) over time. In this description the samples with the same time index or sequence number collectively describe the object pose at that specific time frame.

Point-based (sample-based) 3D model representation, rendering, and manipulation have been extensively studied during the past decade [62, 88]. These representations bring several advantages to the rendering stage, including the possibility of progressive rendering at different levels of detail, early culling decisions, and dense representation of spatial data via hierarchical coding and quantization. Together, these factors all allow rendering at real-time, interactive frame rates even for large models composed of tens of thousands or millions of samples. In this and the next two chapters we study these representations and introduce improved techniques, data structures, and algorithms for hierarchical representation and interactive 3D

rendering of both still and animated point-based 3D models. Specifically, in this chapter we review major works in point-based rendering, and briefly explain different techniques that are used for the representation, visualization, and manipulation of point-based models. We also discuss the advantages and challenges of creating balanced hierarchies, as this will be a major part of our research focus in the next two chapters (Chapters 7 and 8) and the basis of the improvements we have made in this area of research.

6.1 Point-Based Three Dimensional Rendering

In general, there are two paradigms for rendering 2D images from 3D scene data: image based rendering (IBR) and geometry based rendering (GBR). In the GBR paradigm the initial scene is described in terms of geometrical surfaces with different properties such as color, transparency, and reflectance. Conventional rendering techniques are used to synthesize 2D views (projections) from a set of primitives that accurately or approximately represent the object's geometry. In the IBR paradigm the initial scene is described in terms of a set of images that are captured from a 3D scene [107]. These images can be considered as the output of a plenoptic function for some specific range of values assigned to its input variables. Synthesis of a particular view is achieved through reconstruction of a slice of the plenoptic function from the available 2D images [107]. The IBR paradigm has some advantages over the GBR approach. For example, there is no need for construction of a geometrical 3D model. The models can be directly sampled from a real 3D scene using, for example, a CCD camera. However, it is not clear how some operations such as changing the shape of the objects and light conditions can be elegantly performed in an IBR approach [107]. Moreover, the synthesized views may not provide the expected rendering quality.

Point-based representations and rendering techniques (PBR) bring the advantages of both IBR and GBR paradigms to the rendering stage [107]. On one hand, based on the above explanations, 3D points representing an object can essentially be understood as the samples of a plenoptic function, often enabling direct and simplified capturing of the 3D model. In fact, these samples can be practically estimated/reconstructed using range scanners and/or several views of the object captured from different viewpoints surrounding the object. On the other hand, these 3D points, as the simplest geometric primitives, collectively and implicitly describe the geometry of a 3D object in a simplified way, providing a view-independent rep-

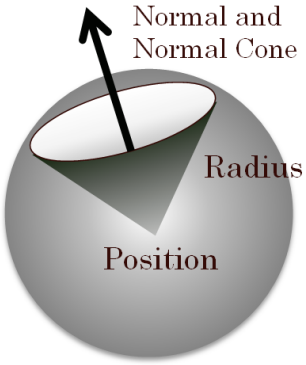


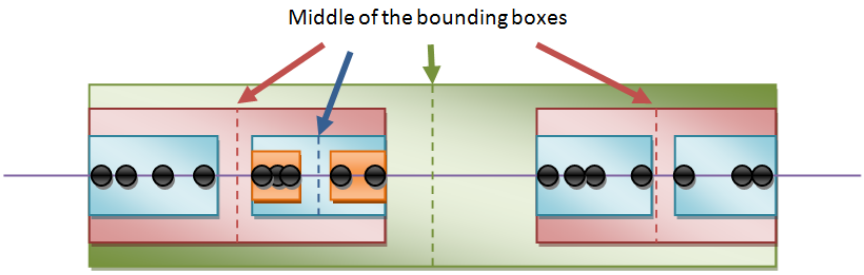
Figure 6.1: A point sample and some of its properties: position, radius, normal, and normal cone.

resentation of the object [46]. In this regard, there is no need to keep information of connectivities between the descriptive elements of the scene. The 3D points are simply assumed to be connected if they are close enough to each other. In addition, point-based representations conveniently accommodate rendering at different levels of detail through the use of appropriate hierarchical structures, and possibly by applying different culling techniques for fast, quality visualization of the object. In practice, the 3D point samples, in addition to their 3D position coordinates, carry some other attributes. These may include point normal and radius, color, transparency, reflectance properties and so forth (see Figure 6.1).

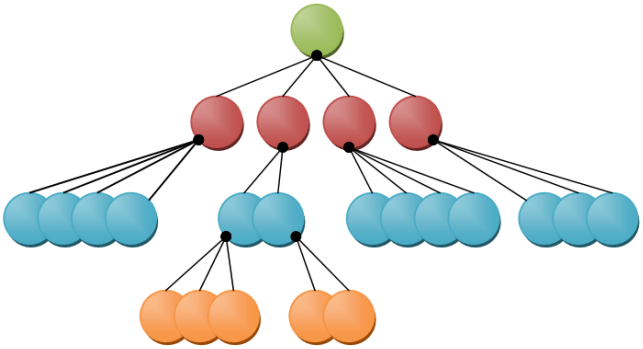
The idea of using points as rendering primitives dates back to the work of Catmull in 1974 [21, 36]. In 1985 Levoy and Whithed introduced the notion of point as a general meta-primitive for the purpose of separating modeling geometry from the rendering process and showed how the geometrically defined objects, curved or flat, can be transformed into points [69]. However, major studies on PBR techniques have been undertaken during the past decade, begining with the work of Grossman and Dally in 1998 [36, 88, 46]. The idea of using a multi-resolution hierarchy in PBR for the first time was introduced in 2000 by Rusinkiewicz and Levoy in QSplat [36, 85] and at the same time by Pfisher *et al.* in Surfels [36, 82]. In fact, QSplat and Surfels are two major leading works in the area of multi-resolution PBR.

6.1.1 QSplat Multi-resolution Representation

In QSplat the model is represented in form of a 4-*d* hierarchical tree of bounding spheres in which each level of the hierarchy represents a level of detail. Each node of the tree contains the sphere center (vertex position), a normal, the width of the



(a) Recursive break-down of the points w.r.t the middle of the bounding box at each level.



(b) The corresponding 4-*ary* tree structure for the breakdown depicted in (a).

Figure 6.2: Pictorial representation of QSplat build tree process using one-dimensional points.

normal cone and some other optional attributes such as color. The hierarchy is built by recursively splitting the bounding box of the vertices into halves along its longest dimension until sub-partitions containing 4 or less elements are achieved. Then these groups of four or less vertices are repeatedly combined to form the vertices in the upper level. In this process the position of the upper level vertices (internal nodes of the tree), and also their normal and color, are calculated as the average of the properties in their children. Figure 6.2a illustrates this recursive process using one-dimensional data. The tree corresponding to the break-down in Figure 6.2a is depicted in Figure 6.2b.

The radius of the parent nodes is calculated so that the parent sphere contains all of its children, and the width of the normal cone is conservatively calculated to guarantee proper visibility of the geometry from the viewpoint of the camera (user) at lower levels. At the highest level, the process simply creates a single sphere that can be considered as the bounding sphere of the whole object.

In writing the hierarchy into a file the position and radius of the vertices are quantized relative to their parent radius. Normals and colors are quantized independently. During the rendering the 3D vertices are progressively reconstructed

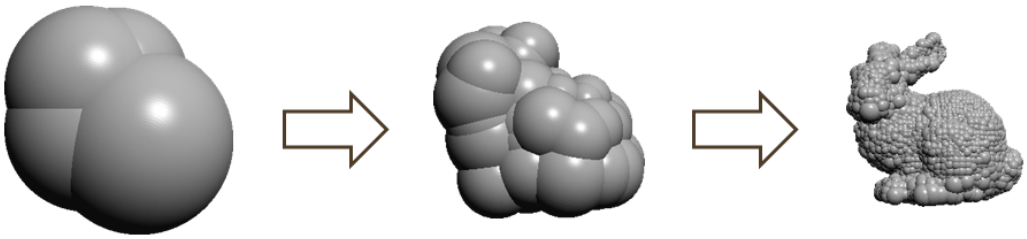
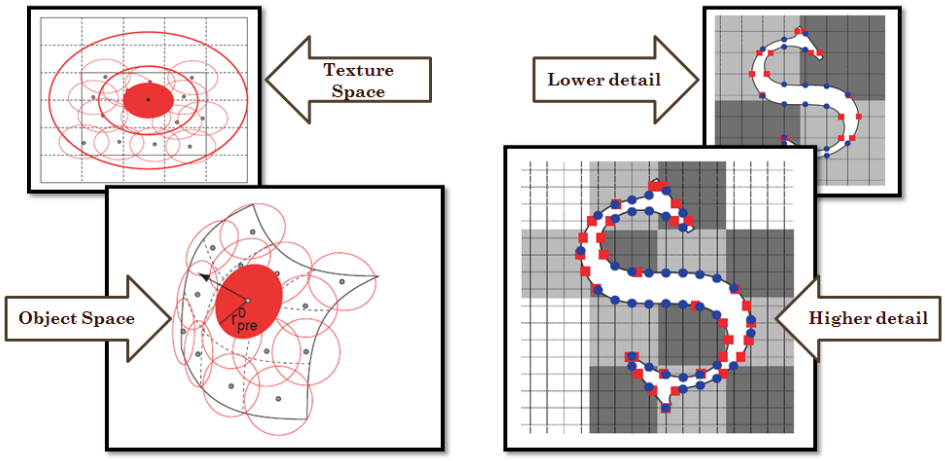


Figure 6.3: Spherical visualization of point samples at different levels of detail for the Bunny 3D model (Figure is adopted from [85]).



(a) Sufels: Surface elements.

(b) Layered depth cubes (LDC), 2D representation

Figure 6.4: Surfels and two dimensional representation of layered depth cubes (Figures are adopted from [82]).

(decoded) starting from the root toward the leaves. The level of detail, *i.e.*, the level of progress toward leaves, is decided by the screen resolution and proportional to the available computational power. See Figure 6.3 to see the spherical visualization of the samples at different levels of detail for the Bunny model.

6.1.2 Surface Elements and Layered Depth Cubes

Surfel, which is an abbreviation for surface element, is defined as a zero-dimensional n-tuple with shape and shade attributes that locally approximate the surface of an object. Like to the spheres in QSplat, surfels have the position and the normal. However, instead of using a sphere for representing the vertex, a circle, which is centered at the surfel position and is located on its tangent plane, is used. These circles, called tangent disks, are used to determine the surfel neighborhood color

or texture properties (see Figure 6.4a). The hierarchy is built using layered depth cubes (LDC) (see Figure 6.4b). During the rendering the LDC tree is traversed from the top (the lowest resolution) to the bottom (the highest resolution). The bounding box of the LDC blocks at each level can be used for frustum-culling and, as in QSplat, the visibility cones can be used for back-face culling [82].

6.1.3 QSplat Variations and Optimizations

Several other hierarchical representations are introduced that are inspired in some respect by QSplat or that try to achieve improvements using some properties of the QSplat structure. In [22], a hybrid point and polygon rendering system called POP is introduced. POP uses the same structure as QSplat but allows switching to polygon rendering at the lowest level of the hierarchy. This ensures the quality of the rendered object at closer distances (*e.g.*, when a user brings the object closer to the camera or zooms into it). PMR, point-to-mesh rendering, is another rendering system that implements a hybrid point-polygonal rendering [36]. However, the hierarchy is built up in a totally different way. PMR builds a multi-resolution hierarchy for both points and triangles at all levels. Thus, it has access to the multiple resolution of triangles as well as of points. In PMR, unlike QSplat, the point hierarchy is created following a feature-based approach introduced in [34] for decimating samples for the purpose of mesh simplification. Roughly speaking, the method calculates the maximum distance of the 3D point to its neighboring samples d_r (d_r may also considered as the radius of the point) and its approximate distance from the medial axis of the object d_m. Then, if the ratio of d_r to d_m is smaller than a threshold τ, the 3D point is considered an oversampled point and is decimated. PMR creates different levels of the hierarchy by adjusting the value of the threshold τ. The triangle hierarchy is built using the Delaunay triangulation method introduced in [35] for shape reconstruction. The method is used to equip each point in each level of the point hierarchy with a set of triangles that approximate the surface of the object at that resolution level. During the rendering the radius of the point d_r is used to decide between rendering a point or a set of triangles attached to the point. In [70] a case study on rendering massive data sets is presented that again benefits from a hybrid polygonal and point-based rendering. The work is significant in the method of handling large data sets. This is accomplished by partitioning the data set into square chunks with some overlapping margins, after which each chunk is processed independently.

In [95] and [96] the authors investigate the geometrical properties of the QSplat structure and the distribution of child attributes (position and normals) relative to those of their parents. Specifically in [95] child spheres sorting and the parent normal cone are used to reduce the set of valid indices for children's positions and normals respectively. Applying these reduction techniques, they can achieve lossless compression of QSplat data up to 60 percent of the original storage requirement. In [96] the positions of the children are transformed to a local coordinate system defined by the position and normal of the parent. Then, benefiting from the more compact representation of the positions of the children in the local coordinate system, vector quantization (instead of scalar quantization) is applied on the groups of children to provide a compact, adaptive distortion-rate representation of the QSplat data.

PBR techniques are also applied in rendering volumetric data. TetSplat is an algorithm which is proposed for real-time rendering and volume clipping of point samples obtained from tetrahedral meshes [77]. In this algorithm the input mesh is divided into two parts called shell and solid. The hell is the exterior or the visible surface of an object and the solid is the interior part or invisible tetrahedrals assuming the volume is an opaque object. Two hierarchical trees are created for both shell and solid parts. The hierarchical tree for the shell is built similarly to the unbalanced quad-tree used in QSplat, but since the solid part is in fact a triangulated 3-manifold, an octree is used to build up the tree. No normal is stored for the samples of the solid part. Shading of the interior part is derived from a CSG probe that is used for volume clipping during the rendering process.

6.1.4 Other Multi-resolution Representations

There are several other works on PBR which use many different techniques for building up the hierarchy tree and rendering points. In [18] a representation is introduced which automatically balances the sampling density and quantization accuracy. Such a representation is achieved by first fitting a regular 3D grid to the 3D samples of the object so that each cell of the grid contains at most one 3D point of the model. Then, the 3D grid is recursively sub-sampled until a single cube, that is the bounding box of the model, is reached. Binary values (0 or 1) are used to show whether or not a cubic section of the space is occupied by some samples. The representation is highly compact, but the need for explicit addressing and correction of the leaves' position reduces its size efficiency.

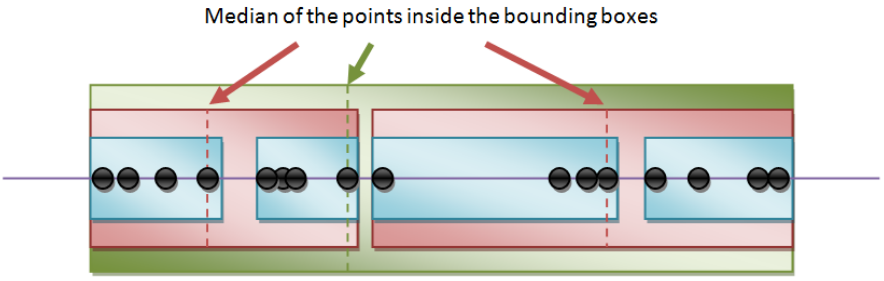
In [43], a layered approach is proposed that organizes the points into a hierarchy of nearly equally sized point clusters so that the final multi-resolution structure has the same number of points as the input. This is obtained by uniformly selecting a small subset of samples as the root of the hierarchy. The remaining samples are spatially divided into halves, and the process is repeated on each half until a predefined size limit is met. In [80] a hierarchy of the plane patches is created by fitting a plane to the positions of the children at each level. The child planes are encoded with respect to their parent plane using the distance of three points that are appropriately selected on the parent plane and their projections along with the parent normal on the child plane.

In [106] a point-based method for rendering large tetrahedral meshes is introduced which can potentially be used on non-conforming meshes as well. The paper introduces a method for sampling the tetrahedral elements of the mesh one by one so that the sample set provides a good approximation of the underlying scalar field. An efficient stratified decimation is used to decimate samples when the renderer is initialized. PBR techniques are used in combination with ray tracing as well [55, 83], and finally, PBR techniques are available for single-pass or multi-pass multi-view rendering [53, 54] (see also Section 6.2).

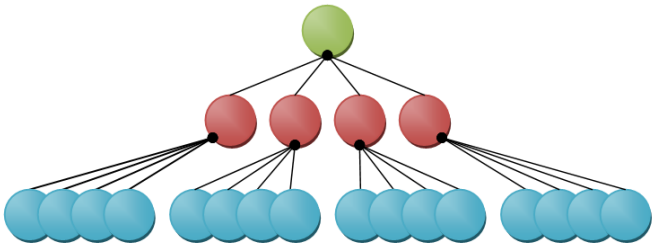
6.1.5 Balanced Hierarchical Representations

All the above-mentioned methods lead to imbalanced structures. In recent years there has been a tendency toward balanced representations, as it is generally expected that a balanced hierarchy results in faster rendering speed (for the same structure) [58], lower memory requirements [57], and more symmetrical progressive rendering. A balanced k-dimensional tree can be achieved by splitting the points based on the median of their coordinate in the splitting dimension [57]. Figure 6.5a represents the breakdown process according to this scheme for one-dimensional data, and Figure 6.5b shows the corresponding balanced tree. Comparing the trees in Figure 6.2 and Figure 6.5, it can be seen that a balanced tree has a more dense representation for the same collection of points.

In practice, a complete balanced tree often cannot be created since the number of vertices is not usually a power of the tree branching factor. A left-balanced tree may be considered [57], but it does not completely use the whole address space of the tree. Hubo *et al.* [55] use a *kd*-tree that stores the quantized value of the split plane position at each internal node. The method is, in essence, similar to the com-



(a) Recursive break-down of the points w.r.t the median of the points at each level.

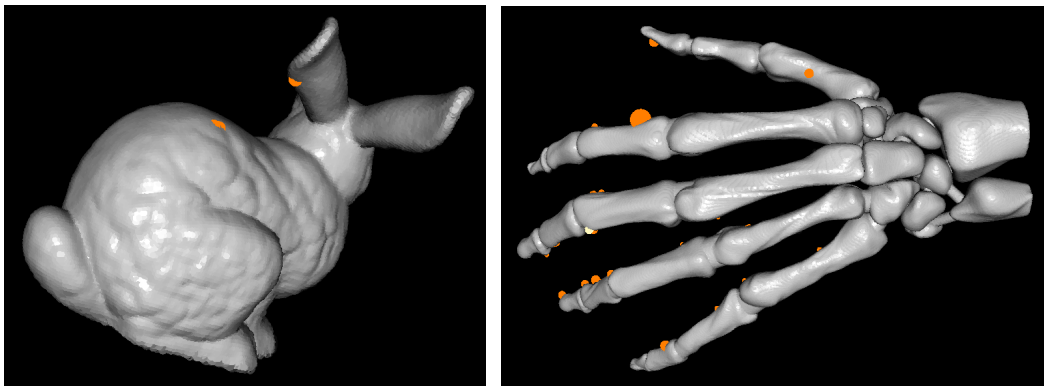


(b) The corresponding 4-dimensional balanced tree structure for the break-down depicted in (a).

Figure 6.5: Pictorial representation of building a 4-dimensional balanced tree using one-dimensional points.

pact representation in [18], but the authors need to keep track of the splitting point (median) in a balanced representation. Kalaiah and Varshney [58] introduce a statistical representation (*vs.* deterministic representation) based on principal component analysis. PCA is used to estimate mean and variance of point positions and their attributes. The hierarchy is built using *k*-means clustering. Mahalanobis distance or alternatively Euclidean distance is used for clustering. In either case, especially for the Mahalanobis distance, their method may not achieve a well-balanced structure. They switch to Euclidean distance if the Mahalanobis distance leads to an extremely imbalanced tree structure. The mean, variance, and principal components at all levels are quantized and stored. For the rendering, these estimated Gaussian distributions are qausi-randomly sampled and splatted to the screen. Goswami *et al.* [44] propose a balanced multi-way *kd*-tree with an adaptable branching factor. An adaptive clustering technique is introduced that uses the position and normal attributes to disseminate the points among the children.

Apart from the number of samples, creating a balanced *k-ary* tree also mandates distributing equal number of constructive elements of the model, *i.e.* points, triangles and so on, among a specific number of groups while the proximities inside each group also need to be preserved. The constraint of having the same number



(a) Bunny (b) Hand

Figure 6.6: Artifacts (highlighted points) resulted from inappropriate grouping of points in balanced partitioning.

of primitives in each group increases the possibility of grouping points that are not close enough to each other. This can be seen in Figure 6.5a for our sample one-dimensional data, where the point in the middle is grouped with three other points that are far from it. Of course, the probability of grouping the points that are not close enough to each other increases with the use of 3D data. This may lead to considerable error in hierarchical quantization, and may later cause serious artifacts in model restoration and rendering. Figure 6.6 shows some of these rendering artifacts in the case of QSplat, where the median of the samples is used to break the samples down. These rendering artifacts appear in the form of points (spheres) that are dissimilar in size and inaccurate in position relative to their neighboring points. Some of these artifacts are highlighted in Figure 6.6 for better visibility. As perceived, these artifacts may occur even with some simple models such as the Bunny (Figure 6.6a). The problem intensifies with more topologically complex objects such as the Hand model (Figure 6.6b). This phenomenon is not usually observed in the case of imbalanced hierarchical trees since these structures often asymptotically follow the geometry of the object. Clustering techniques may be applied to put nearby points in the same group [44, 58], but as mentioned above these techniques may not lead to completely balanced structures [58] or may not properly work on all 3D surface models.

6.2 Stereo Visualization

There are two different approaches to (multiview) stereo visualization: multi-pass rendering and single-pass rendering. Multi-pass methods repeat the underlying single-view rendering algorithm with appropriate view-dependent parameter settings for N different passes where N is the number of views. The result of N passes may then be masked and composed in different ways depending on the targeted stereoscopic display device. A multi-pass rendering algorithm that does not exploit the similarities between different views can take N times longer than a mono-view rendering algorithm [38]. In this case, reducing the whole processing time can be accomplished by improving the efficiency of the underlying single-view rendering algorithm. There are also some acceleration methods which try to manipulate and transform the primitives projection on a single, directly rendered view to the corresponding projections on the other desired views assuming that the properties of the corresponding primitives in different views are the same [50, 66, 101, 5]. However, the speed improvements obtained by these methods is generally at the cost of image quality. In particular, for the case of auto-stereo, where several views are needed, these methods may lead to more and stronger artifacts [53]. As an alternative approach, the actual rendering can be done for more than one, but a subset of the total number of required views. Then, the desired in-between views may be obtained by applying appropriate image-based rendering techniques. Single-pass multiview rendering methods are more recent techniques that try to simultaneously generate all views that are needed for the 3D display in a single pass over the 3D data. Some works in this area exploit the programming capabilities of modern GPUs for direct calculation of multiple views on a per-fragment basis [54, 53]. Nevertheless, even in these methods the basic per-fragment operations that are required for generating multiple views remain proportional to the number of views.

6.3 Sampled 3D Object Deformation and Manipulation

There has also been substantial research into the methods of manipulating and deforming sampled object representations, including point-based 3D models [23]. Deformations are often achieved by applying some empirical (no or little physics involved) or physically-based deforming models on sampled representation of fluid, solid, or elastoplastic objects [42, 81, 3], and may use a combination of different

preprocessing and manipulation steps. Deformation models can be considered as the integral part of the computer animation, *i.e.*, modeling and rendering temporal behavior of the object. Depending on the model complexity and size of the object, these models may not often be useful within an interactive environment. However, they can be used to generate a series of object poses (and properties) from motionless 3D objects. These simulations can be considered a valuable source of spatio-temporal samples describing a dynamic 3D object.

Chapter 7

Interactive Visualization of Still Point-based 3D Models Using Balanced Hierarchies

Balanced structures can generally be advantageous in terms of memory requirements due to more efficient use of the tree addressing space, as well as to the possibility of implicit representation; *i.e.*, there is no need to explicitly store information related to the tree structure itself such as the number of children of a node and their address. In addition, subject to the appropriate dissemination of the samples among children, they can also lead to a more harmonized progressive rendering which makes the approach more convenient for interactive and/or remote rendering. However, as already mentioned, it is in practice very improbable to have a number of samples appropriate for creating a complete multi-way balanced tree structure, and it is not quite obvious how exactly to disseminate samples among children. In this chapter we address the first problem by introducing a structure called a *Multi-section* tree, which allows a multi-way balanced tree to be built for any arbitrary number of samples very close to the structure of a complete tree of the same branching factor. We also introduce algorithms and techniques for the latter problem, *i.e.*, appropriate distribution of samples within the balanced tree structure.

7.1 Balanced Multi-section Tree Structure

The *Multi-section* tree, hereafter also MSTree, can be considered as an extension of *bisection* tree [45] with an arbitrary branching factor $m = 2^l$ (for *bisection* tree

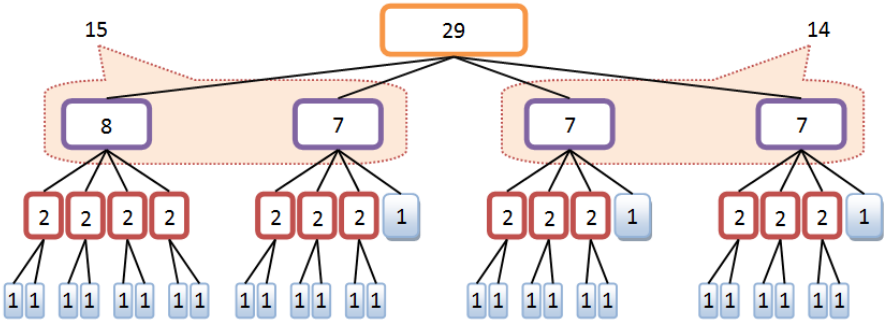


Figure 7.1: Balanced *multi-section* tree: the numbers of vertices assigned to the siblings are either equal or differ at most by one.

$m = 2$). The structure is built by recursively dividing the set of vertices assigned to each node of the current level into m subsets which are assigned to the m children in the next level, so that the numbers of vertices in all subsets are equal or differ by at most 1. As a result, all of the internal nodes at all levels get the maximum possible, literally equal number of children. In following subsections we describe the recursive process of building the MSTree structure and explain how the tree structure can be traversed implicitly.

7.1.1 Constructing the MSTree Structure for an Arbitrary Number of Vertices

Algorithm 1 presents the recursive process of building a generic *multi-section* tree with a branching factor of $m = 2^l$ for an arbitrary number of multi-dimensional samples. The recursive process stops if the breakdown reaches a partition of size smaller than or equal to m. Otherwise, the *bi-partition* procedure is repeatedly applied to divide the input into m sub-partitions. Here, *bi-partition* is a general partitioning or clustering procedure that divides its input P into two equally sized partitions P_1 and P_2 assuming that it devotes one more sample to P_1 if the number of samples in P is odd. In this way, the numbers of samples assigned to each of the m sub-partitions and consequently the numbers of samples assigned to the siblings at each level (the number of leaves beneath all siblings) differ by at most one. As we will see, this helps in distinguishing the leaves from the internal nodes without explicit labeling, even if they are shuffled with each other at the penultimate level. In addition, this allows the whole tree structure to be stored on a disk without using any explicit addressing at all levels.

Algorithm 2 presents a simpler version of Algorithm 1 for the special case of $m = 4$ which is easier to read. Figure 7.1 shows an example of applying Algorithm 1 (or Algorithm 2) to break down 29 vertices with a branching factor of $m = 4$. The numerical node labels in this figure show the number of vertices (leaves) beneath each internal node. As can be seen, it is not necessary that the number of vertices be a power of 4 to end up with an equalized tree structure. However, when the number of vertices is close to a power of m, the space available in the last row will be used more efficiently.

Algorithm 1: Recursive construction of *multi-section* tree.

$n = \textbf{\textit{mBalance}}(P, m)$
Input: P: Set of points, $m = 2^l$: Branching factor
Output: n: Tree node

if $|P| < m$ **then**
    return ***Combine***(P);
$Q.PushBack(P)$; `// Q is a queue`
while $(Q.Length < m)$ **do**
    $\{P_1, P_2\} =$ ***bi-partition***$(Q.Front())$;
    $Q.PopFront()$;
    $Q.PushBack(P_1)$;
    $Q.PushBack(P_2)$;
for *(all P_i in Q)* **do**
    $v_i =$ ***mBalance***(P_i, m);
return ***Combine***$(\{v_i\})$;

Algorithm 2: *4Balance* build tree algorithm.

$n = \textbf{\textit{4Balance}}(P)$
Input: P: Set of points
Output: n: Tree node

if $|G| < 4$ **then**
    return ***Combine***(P);
$\{P_1, P_2\} =$ ***bi-partition***(G);
$\{P_{11}, P_{12}\} =$ ***bi-partition***(P_1);
$\{P_{21}, P_{22}\} =$ ***bi-partition***(P_2);
return ***Combine***(***4Balance***(P_{11}), ***4Balance***(P_{12}), ***4Balance***(P_{21}), ***4Balance***(P_{22}));

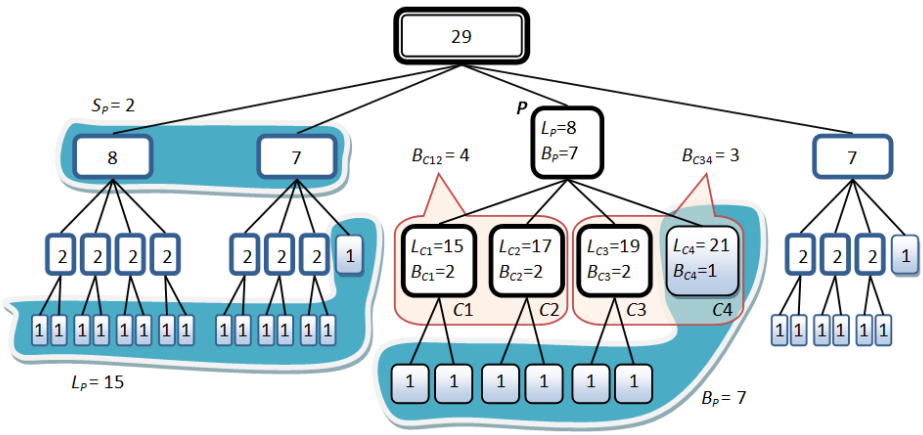


Figure 7.2: Calculating various parameters necessary for MSTree traversal; S_P: number of siblings in the left hand side of P, L_P: number of leaves in the left hand side of P, and B_P: number of leaves beneath P.

7.1.2 General MSTree Traversal Scheme

Assuming that the MSTree structure is stored in (main) memory in a breadth first order (see Figure 8.2a), the tree can be traversed following a recursive equi-division process similar to the recursive equi-partitioning method that was used in building the tree structure. Figure 7.2 gives a pictorial description of the different parameters involved in this process and shows how they are calculated. For a parent (internal) node P, the number of its immediate children is calculated simply as the minimum of m, the tree branching factor, and B_P, number of leaves beneath node P (for the root node R, B_R is the total number of vertices). In other words, each internal node P will have m or less number of immediate children. For each immediate child C of P, number of leaves beneath C, B_C, is calculated from B_P by repeatedly dividing B_P and subsequent halves by 2 until size of all immediate children is found. For example, B_C for the children of node P in Figure 7.2 is calculated as:

$$B_{C12} = \lfloor B_P/2 \rfloor + 1 = 4$$
$$B_{C34} = \lfloor B_P/2 \rfloor = 3$$

$$B_{C1} = \lfloor B_{C12}/2 \rfloor = 2, \qquad B_{C2} = \lfloor B_{C12}/2 \rfloor = 2$$
$$B_{C3} = \lfloor B_{C34}/2 \rfloor + 1 = 2, \qquad B_{C4} = \lfloor B_{C34}/2 \rfloor = 1 \qquad (7.1)$$

In practice, B_C values can be directly obtained from B_P by dividing it by the tree branching factor m, and then through appropriate handling (adding up and rounding) of the fractional part of the division. B_C can be used to distinguish a leaf from an internal node: a tree node C is a leaf simply if $B_C = 1$.

Other parameters for recursive traversal of the tree are calculated as follows:

- The number of leaves to the left hand side of C, L_C, is calculated as the number of leaves in the left hand side of the parent node P, L_P, plus the summation of B_C for all children of P on the left hand side of C.
- The number of siblings to the left hand side of C, S_C, is calculated as $S_P * m + i$ where i is the number of immediate children of P on the left hand side of C.

Knowing L_C and S_C, it is easy to determine S_C^i and S_C^v, the number of siblings on the left hand side of C that are internal node and leaf, respectively ($S_C = S_C^i + S_C^v$). From there the children offset of C, O_C, can be calculated as:

$$O_C = O_L + S_C^i * iSize + S_C^v * vSize \tag{7.2}$$

where O_L is the offset of level L, and $iSize$ and $vSize$ are the size of the internal nodes and size of the leaves, respectively. $iSize$ and $vSize$ can potentially be different, making it possible to store different/additional information in leaves of the tree.

7.2 Using MSTree in Balanced Representation of Still 3D Objects

Algorithm 1 refers to a general partitioning procedure named *bi-partition* that divides input vertices into two sets of the same size. Different partitioning or clustering schemes can be applied here. However, the technique should be able to appropriately distribute the points among the clusters so that the proximity of the points inside the partitions be preserved and therefore high quantization errors, and hence noticeable rendering artifacts similar to those appearing in the case of median-based partitioning (see Figure 6.6), can be avoided. In general, as explained in Section 6.1.5, establishing such a well-balanced hierarchy remains a challenge as clustering techniques that work for a class of models may not generate desirable results for

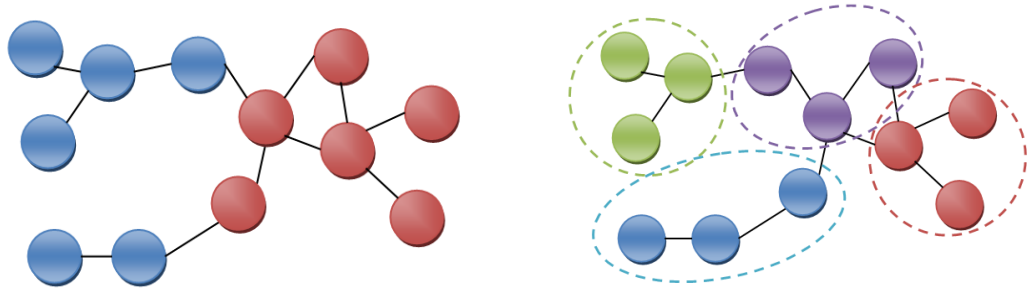
other structurally different classes of objects. Here we introduce two algorithms for distributing samples within an MSTree structure.

The first algorithm, Algorithm 3, is a graph-based partitioning scheme in which we try to preserve the proximity of samples within sub-partitions by keeping them as connected components of a graph to the greatest extend possible. The algorithm is based on recursively applying a type of graph bi-partitioning that tries to preserve connectivity of the partitions instead of minimizing the number of edge cuts. The resulting partitions are not necessarily located on two sides of a splitting plane but resemble a patch of the neigbouring points on the surface of the model. As a result, their points are spatially close enough together to avoid the aforementioned quantization errors and rendering artifacts.

The second algorithm, Algorithm 5, is a two-phase approach to the problem, *i.e.*, a bottom-up grouping using Algorithm 4, followed by a top-down recursive splitting process, resulting in a two-layer structure. The initial bottom-up grouping keeps neighboring points together at the first stage, avoiding the necessity of applying a perfect, intricate clustering algorithm. Again, considering the whole surface of a 3D object as a graph, we try to partition the surface into a set of small, nearly equally sized connected components or patches. Then the MSTree structure is used to build a set of balanced hierarchies (subtrees) for each of these groups. The main MSTree is created over all group representatives (subtree roots) [10]. As we will see, this two-layer balanced representation results in a very compact representation (as low as 25 bits per sample radios, position, and normal) compared to the first algorithm, Algorithm 3, and also compared to the other, state-of-the-art algorithms.

7.2.1 Graph-based Dissemination of Vertices

Assume that the whole surface of a 3D object is represented as a graph $G(V, E)$, where V is the set of vertices of the model and E represents the set of edges connecting the neighboring vertices. A 3D mesh obtained by applying Delaunay triangulation [35] on the object samples may simply be thought of as a graph representation of the model. Having this representation, we turn the problem into a graph partitioning problem with the specific constraint of preserving the connectivity inside each partition while trying to employ the geometric properties of the shape as well. The intuition behind this approach is the close relationship between proximity of vertices in a 3D surface model and connectivity of the vertices in the corresponding graph representation. Therefore, if we keep the sub-partitions internally connected



(a) A sample graph that cannot be partitioned into two same-size connected components.

(b) Partitioning of the graph in (a) into four same-size connected components.

Figure 7.3: Partitioning a graph into same size connected components.

we, in fact, group the vertices that are close to each other.

Contrary to the classical graph partitioning problem that looks for the pieces that have the fewest connections between them (or equivalently looks for the fewest number of cuts) [60], in our graph-partitioning it does not matter how many edges are cut. Enforcing the connectivity constraint inside sub-partitions may, however, lead to cases where the problem does not have a solution at all. Figure 7.3a shows an example of such a graph, which cannot be separated into two connected pieces with the same number of nodes. Nevertheless, this will not be a major issue, at least in the higher levels of the breakdown process, considering the number of vertices used for a typical 3D model representation.

The graph partitioning problem is called a *graph bisection problem* when the number of targeted partitions is two [41]. A graph can be partitioned into $m = 2^k$ components of (approximately) the same size by recursively repeating the bisectioning algorithm to the subcomponents for k times (see Statements 4 to 9 of Algorithm 1). It should be clarified that in general such a recursive algorithm may fail to reach to the final solution, even if such a solution is possible. For example, the graph in Figure 7.3 can be partitioned into four connected components (see Figure 7.3b) but a recursive bisectioning fails to achieve this solution since it cannot divide the whole graph into two connected components in the first step. Nevertheless, in practice the recursive bisectioning is a good approach to a multi-partitioning problem.

The pseudo-code of our bi-partitioning algorithm, called *ConnBiPart*, is shown in Algorithm 3. In this algorithm *MedPart* refers to the median-based partitioning. *ConBiPart* uses *MedPart* in two ways. Firstly, *MedPart* is tried on all three dimensions of the bounding box D and the connectivity of the resulting partitions is

checked to see whether or not it can solely divide the graph into two connected components (Lines 5 to 9). Secondly, it is used whenever *ConnBiPart* cannot achieve two equally sized connected components. This includes the case where P itself is not connected, a possibility which is checked before proceeding to any further step (Lines 2 to 4). The algorithm goes to the next phase, if applying *MedPart* on all dimensions does not lead to a solution. It starts by initializing P_1 and P_2 with two vertices, namely near end v_n and far end v_f, which are the vertices located on the two ends of the longest dimension of the bounding box of P. Starting from the near end v_n (alternatively can be the far end), the algorithm tries to grow P_1 by repeatedly attaching the neighboring vertices that are *closer* to that end, until a connected set of vertices P_1 of half the size of the whole graph P is achieved. At this point, the remaining undecided vertices may form one or more connected components.

Algorithm 3: *ConnBiPart* bi-partitioning algorithm.

$\{P_1, P_2\}$ = ***ConBiPart***(P)
Input: Partition P
Output: Sub-partitions P_1 and P_2 so that $|P_1| = (|P| + 1)/2$ and $|P_2| = |P| - |P_1|$

1 Let $D = [d_1, d_2, d_3]$ be the bounding box of P so that $d_1 \geq d_2 \geq d_3$;
2 **if** *P is not connected* **then**
3 **return** $\{P_1, P_2\}$ = *MedPart*(P,d_1);
4 **for** *all d in D* **do**
5 $\{P_1, P_2\}$ = *MedPart*(P,d);
6 **if** *both P_1 and P_2 are connected* **then**
7 **return** $\{P_1, P_2\}$;
8 v_n = the vertex with minimum projection on d_1 (near-end);
9 v_f = the vertex with maximum projection on d_1 (far-end);
10 $P_1 = \{v_n\}, P_2 = \{v_f\}$;
11 **repeat**
12 Add $v \in P - (P_1 \cup P_2)$ to P_1 if v is adjacent to some vertices in P_1 and is *closest* vertex to v_n;
until $|P_1|$ = $(|P| + 1)/2$ *or no new elements can be added to* P_1 ;
13 Repeatedly add $v \in P - (P_1 \cup P_2)$ to P_2 if v is adjacent to some vertices in P_2;
14 Add all $v \in P - (P_1 \cup P_2)$ to P_1;
15 Balance number of vertices in P_1 and P_2 by *growing* P_2 toward P_1 if $|P_1| > |P_2|$ or P_1 toward P_2 if $|P_2| > |P_1|$;
16 **if** *succeeded to balance without disconnecting P_1 and P_2* **then**
17 **return** $\{P_1, P_2\}$;
18 **return** $\{P_1, P_2\}$ = *MedPart*(P,d_1);

One of these components will be adjacent to v_f which is already tagged as P_2. The vertices belonging to that component are all tagged as P_2 (Line 17) and all other undecided vertices (if any) are appended to P_1 (Line 18). Finally, the algorithm tries to balance the number of elements in the two partitions by growing P_2 toward P_1 (if $|P_1| > |P_2|$, which is usually the case) or vice versa without losing their internal connectivity. If it does not succeed in maintaining the connectivity constraint it simply switches back to the *MedPart* scheme again.

7.2.2 Balanced Dissemination of Vertices in a Two-Layer MSTree Structure

The main problem with Algorithm 3 is that it does not always generate connected components. Our second algorithm addresses this problem by introducing an initial bottom-up grouping step that keeps the neighboring points together at the first stage, eliminating the necessity for applying a perfect, intricate clustering algorithm.

The bottom-up grouping of neighboring vertices is modeled again as a graph

Algorithm 4: Bottom-up grouping of 3D object vertices.

$\{C_i\} =$ ***GroupObjVerts***$(G(V,E), f)$
Input: Object graph $G(V,E)$, Grouping factor f
Output: Set of connected components $\{C_i\}$

1 $V_g = \Phi;\ V_{ug} = V;\ i = 0;$
2 **while** $(V_{ug} \neq \Phi)$ **do**
3 $\quad C_i = \{v\}, v \in V_{ug} \cap Adj(V_g)$; `// New component`
4 $\quad$ **while** $(|C_i| < f) \wedge (Adj(C_i) \cap V_{ug} \neq \Phi)$ **do**
5 $\quad\quad C_i = C_i \cup \{v\}, v \in Adj(C_i) \cap V_{ug};$
6 $\quad$ **if** $(|C_i| < f)$ **then**
7 $\quad\quad$ **if** $(\exists C_j \in N(C_i)$ *so that* $|C_j| + |C_i| < 2f)$ **then**
8 $\quad\quad\quad C_i = C_i \cup C_j;$
9 $\quad\quad$ **else**
10 $\quad\quad\quad m = \min\{k | C_k \in N(C_i)\};$
11 $\quad\quad\quad$ **while** $(|C_i| < 2f - 1)$ **do**
12 $\quad\quad\quad\quad C_i = C_i \cup \{v\}, v \in Adj(C_i) \cap C_m;$
13 $\quad\quad\quad V_{ug} = V_{ug} \cup (C_m - C_i);$
14 $\quad V_g = V_g \cup C_i;\ V_{ug} = V_{ug} - C_i;\ i = i + 1;$
15 **return** $\{C_i\};$

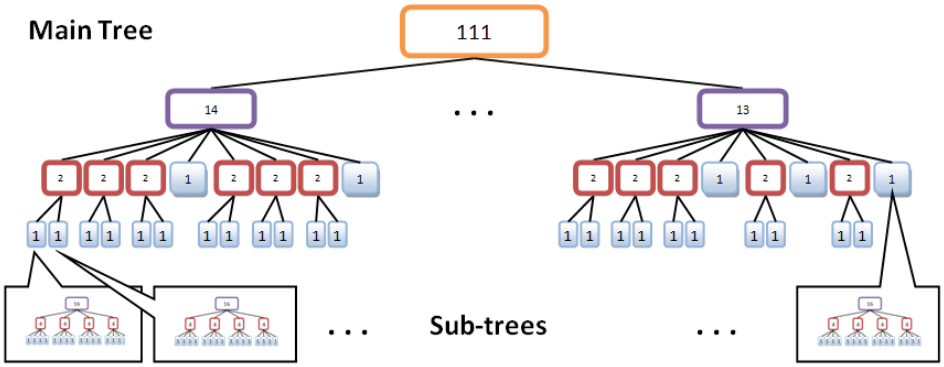


Figure 7.4: The structure of the whole balanced hierarchy.

partitioning problem. Algorithm 4 shows the pseudo-code of this vertex grouping procedure. Here V_g and V_{ug} refer to the set of grouped and set of ungrouped vertices, respectively. $Adj(C)$ is the set of all vertices adjacent to a connected component C and $N(C)$ is the set of all components neighboring C.

The algorithm gets as input a graph $G(V, E)$ representing the surface of the object as before, and a grouping factor, f, that shows the (minimum) size of the connected components (patches) that are going to be created. Starting from an arbitrary vertex $v \in V$, the algorithm repeatedly tries to cut connected components C_i of size f from G (Steps 3 to 5). Whenever a component C_i of size f cannot be achieved (for example when the algorithm is trapped in the object corners) C_i is merged with one of its neighboring components, provided that the total size does not exceed $2f - 1$ (Steps 7 and 8). If all neighboring components fail to satisfy this condition, C_i is merged with part of a neighboring component C_m which has the minimum index m (Steps 9 to 13). Merging C_i with a component which has the minimum index assures that the algorithm will not fall into an endless loop. The remaining part of C_m is returned back to the set of ungrouped vertices V_{ug} for later processing. In practice, the majority of the components created by Algorithm 4 will be of size f.

The whole balanced two-layer hierarchy is created using Algorithm 5. Algorithm 5 first calls Algorithm 4, *GroupObjVerts*, to partition the object surface into a set of nearly equal-size pieces $\{C_i\}$ based on a given grouping factor f. Then by repeatedly calling Algorithm 1, *mBalance*, a subtree with fixed branching factor 4 is created for all these components. The root of each subtree is considered a representative of the corresponding component, and the *mBalance* algorithm is called again to create the main tree over these group representatives using a branching factor m determined by a user. Here, the *mBalance* algorithm uses a simple median-based

approach as its bi-partitioning scheme, but using this simple scheme does not lead to any rendering artifacts because of the initial bottom-up grouping phase. Figure 7.4 shows a sample of the whole two-layer balanced structure for grouping factor $f = 16$ and branching factor $m = 8$.

7.3 Combining Tree Leaves and Nodes

Algorithm 1 uses a procedure named *Combine* which is called within the recursive process to build up the hierarchy. Method *Combine* combines the lower level nodes to form the internal upper-level nodes of the tree. Method *Combine* returns the vertex itself if the partition passed into it has only one vertex (a leaf node); otherwise *Combine* creates a new node which will become the parent node at the upper level. The parent node position is calculated as the average of the positions of its children [85]. Other attributes including normal and color are treated similarly, but considering that our partitioning algorithms keeps the neighboring 3D points together (as demonstrated below), taking advantage of the similarity of these properties within a patch we can more compactly store these information in the internal nodes (especially the internal nodes of the subtrees of the two-layer structure) [10].

7.4 Tree Quantization and Encoding

The hierarchical quantization method in [85] is extended to a multi-accuracy scheme that allows quantizing position and radius of the points with different precisions (9 to 16 bits). Multi-accuracy quantization is implemented using separate quantization/lookup tables. It can also be implemented by mapping all valid combinations at all accuracy levels to the same lookup table at the cost of losing quantization

Algorithm 5: Building the whole two-layer balanced hierarchy.

$R =$ ***TwoLayerHierarchy***$(G(V, E), f, m)$
Input: Object graph representation $G(V, E)$, Grouping factor f, Branching factor m
Output: Root of the hierarchy R

```
C_i = GroupObjVerts(G(V, E), f);
for all C_i in {C_i} do
    r_i = mBalance(C_i, 4);
return R = mBalance({r_i}, m);
```

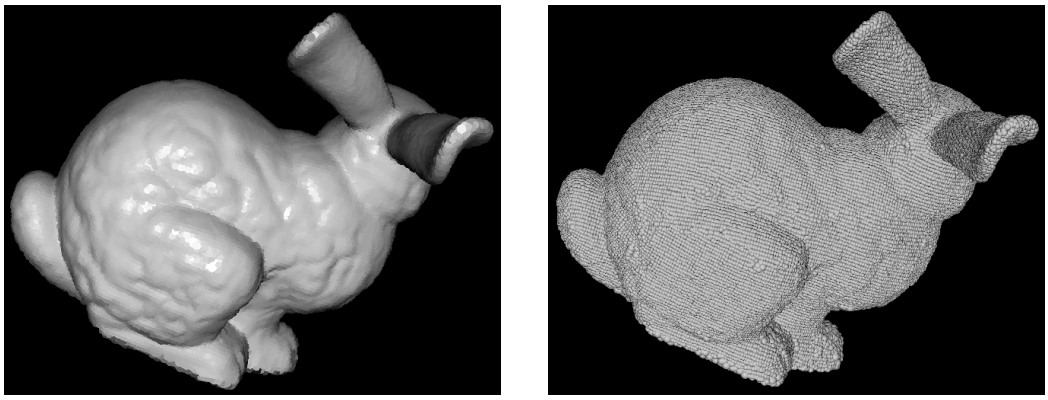
uniformity for lower level accuracies. We have also implemented an improved hierarchical quantization scheme base on a hierarchy of bounding boxes (see Section 8.2) that could be used here to obtain more accurate quantization at the same bit rate.

Dividing the surface of a 3D object into small pieces suggests the plausibility of applying frequency domain encoding schemes based on DCT or Wavelet transform to the compression of point colors, a technique similar to those used in JPEG/JPEG2000 image compression. Point normals are more sensitive to quantization noise and it is better to be quantized independently. Here, normals are quantized based on repeated tessellation of a unit sphere into spherical triangles [18], starting with a regular tetrahedron. 14 bits (7 tessellation levels) are used for storing quantized normals. For the case of a two-layer structure, we still benefit from the directional similarity of the normals in a small neighborhood and factor out the common leading bits of quantized normals within each subtree to devote a smaller number of bits to the normals.

The hierarchy is written into the disk in a breadth-first order. The tree branching factor and the total number of vertices, and some few other parameters' values, such as quantization accuracy, are stored in the header of the file. These are the only information that is needed to traverse the hierarchy (see Section 7.1.2), and there is no need to store any information regarding the tree structure itself. For the case of a two-layer structure, the main tree is stored first followed by all subtrees. Depending on the number of common leading bits of the quantized normals within each subtree, a different amount of memory is dedicated to the normals. In addition, the number of vertices in each subtree can be different. Therefore, there will be some difference in the size of subtrees and due to these differences we need to keep track of the subtrees' offset. This is the only place in the two-layer structure where explicit addressing is used.

7.5 Experimental Results

In the following subsections we present some qualitative and quantitative results of our suggested algorithms, the *ConnBiPart* and the two-layer structure respectively. We compare these results versus some other representations and show their superiority in terms of compactness and accuracy of the representation especially in the case of the two-layer structure. We also briefly discuss the preprocessing complexity of our algorithms.



(a) Point-based rendering (b) Spherical visualization

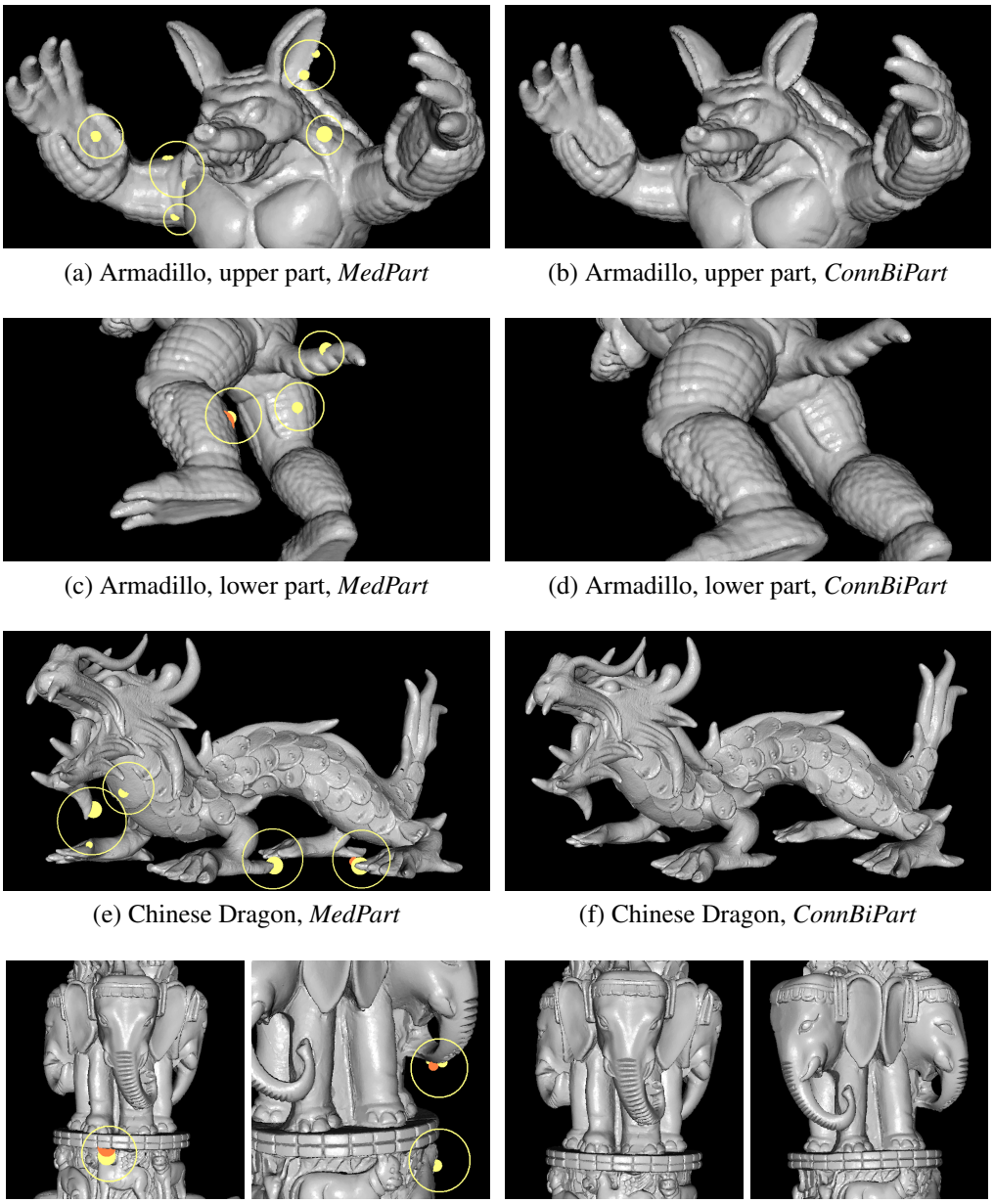
Figure 7.5: Rendering of the Bunny model based on *ConnBiPart* algorithm.

7.5.1 Results of *ConnBiPart* Algorithm

Figure 7.5 shows the result of rendering the Bunny model reconstructed from a representation obtained by applying Algorithm 3, *ConnBiPart*. It can be seen that all of the artifacts shown in Figure 6.6 are successfully eliminated. In fact, for such a simple model *ConnBiPart* does not even need to go further than step 10; balanced partitioning is achieved just by selecting an appropriate splitting axis.

Figure 7.6 presents the rendering results for a set of more complex objects using *MedPart*, simple median-based partitioning, and *ConnBiPart* algorithms. The models include Armadillo, Chinese Dragon, and Statuette. Images in the left hand side of Figure 7.6 show the results for the representations generated by *MedPart*. Some of the significant artifacts caused by median-based partitioning are highlighted in these images. The right hand side of Figure 7.6 shows the corresponding rendering results from representations generated by *ConnBiPart*. The same rendering algorithm is used for both representations. As can be seen all significant artifacts are eliminated in the right hand side images.

Table 7.1 shows the mean, standard deviation, and maximum quantization error for the representations generated by the original QSplat, *MedPart*, and *ConnBiPart* algorithms, respectively. The quantization error for each individual vertex is calculated as the Euclidean distance between the original and the recovered 3D position of each vertex, and the differences are used in calculating mean, standard deviation, and maximum distortion for each model. Table 7.1 also presents the size of the imbalanced structure created by QSplat and the balanced structures created either by *MedPart* or *ConnBiPart*, and the percentage of the saving achieved in balanced



(a) Armadillo, upper part, *MedPart*

(b) Armadillo, upper part, *ConnBiPart*

(c) Armadillo, lower part, *MedPart*

(d) Armadillo, lower part, *ConnBiPart*

(e) Chinese Dragon, *MedPart*

(f) Chinese Dragon, *ConnBiPart*

(g) Statuette, *MedPart*

(h) Statuette, *ConnBiPart*

Figure 7.6: Rendering results for the Armadillo, Dragon, and Statuette 3D models. Left hand side: Rendering from representations obtained from *MedPart*; significant artifacts are highlighted. Right hand side: Rendering from representations obtained by applying *ConnBiPart*; no significant artifact is observable.

representations. Values in Table 7.1 show that for all models the mean and standard deviation of error calculated for *ConnBiPart* is lower than of those calculated for both QSplat and *MedPart*. The maximum quantization error of *ConnBiPart*, except for the two Dragons, is also considerably lower than the maximum error generated

Table 7.1: Quantitative comparison between *MedPart*, *ConnBiPart*, and original QSplat representation.

Alg	Stat	Models						Avg
		Bunny	Armad	Buddha	Dragon	ChDrag	Statuette	
Original	Mean	1.95e-4	0.07084	5.40e-5	6.93e-5	0.01552	0.03325	
	Std Dev	9.15e-5	0.03330	3.55e-5	4.05e-5	0.00772	0.03685	
	Max	5.20e-3	1.82827	0.00102	7.78e-4	1.21989	7.10705	
	Size(KB)	224	1125	3545	2852	23248	32467	10577
MedPart	Mean	1.35e-4	0.05748	3.87e-5	5.66e-5	0.01361	0.03210	
	Std Dev	6.72e-5	0.03041	2.72e-5	4.15e-5	0.00953	0.03897	
	Max	2.76e-3	2.95777	0.00191	3.11e-3	2.76759	6.11213	
	Size(KB)	190	890	2978	2390	17513	26092	8342
ConnBiPart	Mean	1.32e-4	0.05635	3.86e-5	5.62e-5	0.01342	0.03116	
	Std Dev	5.97e-5	0.02486	2.49e-5	3.72e-5	0.00694	0.03576	
	Max	8.86e-4	0.20318	5.56e-4	9.29e-4	1.97643	4.62511	
	Size(KB)	190	890	2978	2390	17513	26092	8342
	Save(%)	15	21	16	16	25	20	21

by QSplat, and significantly lower than the *MedPart* for all 3D models. On the other hand, *MedPart* algorithm in all cases leads to large maximum errors which usually appear as unpleasant outgrowths during rendering (highlighted points in Figure 7.6). In general, *ConnBiPart* leads to better, more smooth rendering while on average more than 20 percent saving in memory is achieved. Depending on the number of vertices and the tree branching factor, a file size up to about 30 percent smaller can be achieved.

7.5.2 Results of Two-Layer Tree Structure

Figure 7.7 shows the rendering result of the Hand 3D model using two-layer balanced representation (*TLTree*). Our two-phase balanced representation does not generate any rendering artifacts for this topologically complex 3D surface (compare with the rendering result in Figure 6.6b).

Figure 7.8 shows different renderings of the Chinese Dragon obtained by *TLTree*, QSplat, and *MedPart*, respectively. Again our method does not generate any artifacts, whereas *MedPart* results in several noticeable artifacts (highlighted splats in Figure 7.8c).

Figure 7.9 shows the close-up of different renderings of the Armadillo obtained from QSplat and *TLTree* at different point position quantization accuracies (11, 13,

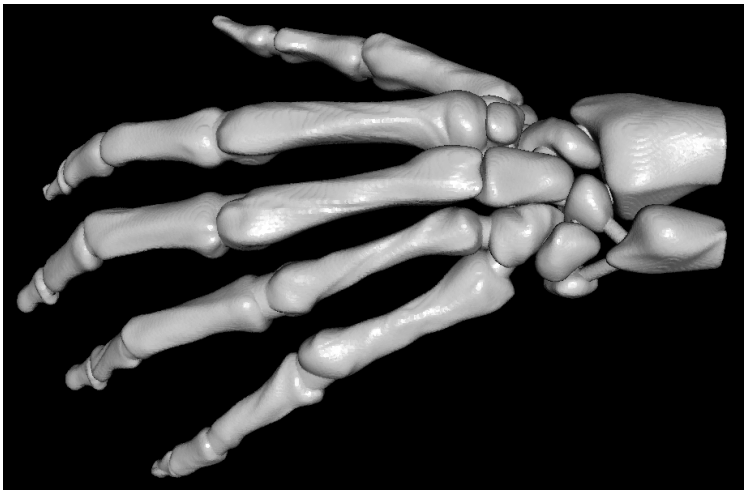
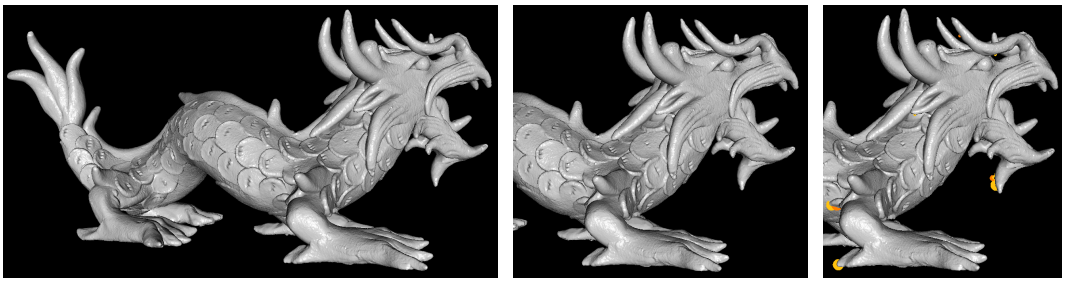


Figure 7.7: Result of rendering from a two-layer balanced representation (*TLTree* representation) of the Hand model. This can be compared with the rendering result in Figure 6.6b.

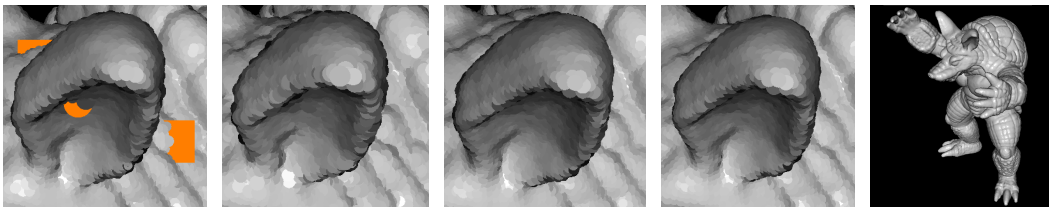


(a) *TLTree*, 13 bits (b) QSplat, 13 bits (c) *MedPart*, 16 bits

Figure 7.8: Chinese Dragon renderings respectively obtained from (a) two-layer representation, *TLTree*, (b) QSplat, and (c) *MedPart*. Major artifacts in (c) are highlighted for better visibility.

and 16 bits). It can be seen that the rendering quality of *TLTree* is comparable to or better than that of QSplat, whereas QSplat fails to appropriately restore parts of the original surface (highlighted splats in Figure 7.9a). Our experience with different models shows that at the same quantization accuracy our method better represents sharp edges than does QSplat. Moreover, our method does not generate any disorder in any quantization precision, though the rendering quality degrades to some extent at lower accuracies (9 and 10 bits).

Table 7.2 shows some of the statistics for different Chinese Dragon representations obtained from the QSplat, *MedPart*, and *TLTree* balancing approach. Again, the quantization error statistics (mean, standard deviation, and maximum error) are calculated from the difference in (the Euclidean distance of) the reconstructed and



(a) QSplat, 13 bits (b) *TLTree*, 11 bits (c) *TLTree*, 13 bits (d) *TLTree*, 16 bits (e) *TLTree*, 16 bits

Figure 7.9: Close-up of the Armadillo renderings obtained from QSplat and *TLTree* representations at different quantization accuracies.

Table 7.2: Accuracy and compactness of *TLTree* versus QSplat and simple median-based balancing, *MedPart*, for Chinese Dragon.

Method	Parameters			Quantization Error Statistics			File Size		
	g	f_{mt}	q (bits)	Mean	Std Dev	Max	Size (KB)	Save (%)	bpp
QSplat	-	4	13	0.0155	0.0077	1.2198	23,248	-	52.76
MedPart	-	4	16	0.0093	0.0061	2.0812	17,513	25	39.74
TLTree	256	4	9	0.0590	0.0284	0.7103	11,034	53	25.03
	256	4	11	0.0309	0.0147	0.3743	12,211	47	27.71
	256	4	13	0.0188	0.0088	0.2816	13,388	42	30.38
	256	4	16	0.0100	0.0047	0.1878	15,147	35	34.37
	64	8	9	0.0599	0.0289	0.8240	10,892	53	24.71

Number of vertices: 3,609,455
Avg sampling distance: 0.1195

g : Grouping factor
q : Pos quantization accuracy
f_{mt} : Main tree branch factor
bpp : Number of bits per point

Table 7.3: Preprocessing time of our method, two-layer tree structure, versus QSplat and simple median-based balancing approach.

Method	g	Model Preprocessing Time (Seconds)		
		Armadillo (172,974)	Blade (879,712)	Chinese Dragon (3,609,455)
QSplat	-	0.457	2.325	9.520
MedPart	-	1.702	9.282	60.271
TLTree	16	3.753	12.651	78.671
	64	1.766	10.838	47.016
	256	1.409	8.000	34.841

the original point positions. The maximum error in simple median-based balancing, *MedPart*, is about 18 times larger than the average sampling distance, even with 16 bits quantization accuracy. This significant error is the source of some noticeable

rendering artifacts, presented in Figures 7.6 and 7.8c. In contrast, our approach, *TLTree*, leads to smaller maximum error even with lowest quantization accuracy. As a result, no significant distortion is observable in the rendering results, while we can achieve file sizes that are more than 50 percent smaller than those of QSplat (see Saving column in Table 7.2) with compactness as low as around 25 bits per point position and normal. To the best of our knowledge this is lower than the values reported in other representations (see [43, 17, 44]), whereas we believe that our method - and in general relative hierarchical quantization - leads to better rendering results than absolute quantizing to the sampling distance used in those representations. Since in our hierarchical quantization point positions are estimated at accuracies lower than the sampling distance (see Mean column of Table 7.2), rendering splats better resemble the original 3D surface of the object.

The complexity of our equi-partitioning algorithm, Algorithm 4, is $O(|V|+|E|)$ where $|V| = n$ and $|E|$ are the total number of vertices and edges, respectively. In the current implementation, we sort vertices along with the splitting dimension to find their median. As a result, the complexity of building the median-based balanced tree is $O(n^2 \log n)$, and the complexity of building the whole two-layer balanced hierarchy is $O(n+|E|+N*(N \log N+f^2 \log f))$ where f is the grouping factor and $N = n/f$ is the number of groups or subtrees. On the other hand, the complexity of creating an imbalanced QSplat tree is $O(n \log n)$. This means that the complexity of the off-line preprocessing phase for creating the two-layer structure stands somewhere between QSplat and *MedPart* Algorithms. In practice, as shown in Table 7.3, the preprocessing time of the two-layer structure, *TLTree*, falls below that of the *MedPart* if the grouping factor f is large enough to affect the whole processing time. It is possible to improve the preprocessing time of both *MedPart* and TLTree by using a subset of vertices, using a linear-time selection algorithm [30], or using both, to find the median. It is expected that these modifications bring the preprocessing time of our algorithms close to that of the QSplat independent of the number of vertices.

Our rendering is optimized for progressive visualization with a rendering speed comparable to that of QSplat. However, since our representation divides vertices into approximately same size groups, much like to the layered representation in [43], which divides the model into approximately equal-size chunks of points, the OpenGL Vertex Buffer Object extension could be used to achieve much faster rendering speeds.

Chapter 8

Interactive Visualization of Animated Point-based 3D Objects

Traditionally, PBR techniques have mostly been used in modeling and rendering of the surface (or sometimes the volume) of still 3D objects. Generalizing this notion, a moving or animated 3D object can be described as a sequence of sampled, object-centered representations of the 3D object over time, rather like a 2D movie can be considered a sequence of sampled (digital) 2D projections. In this representation each set of samples having the same time index collectively describe the 3D object (surface) pose and properties at that specific time index. These representations can be obtained from a wide variety of sources and techniques. For example, such 4D dynamic descriptions can be obtained by sampling analytical object representations [23], directly reconstructed from multi-view video recordings [100], obtained from the data captured by medical imaging devices, or generated by simulating the dynamic behavior of the object using some 3D object deformation models. Having such spatio-temporal 4D representations, it would be very interesting to apply interactive, multi-resolution PBR techniques to the 4D space to simultaneously provide the user with conventional 3D interactions (fly-through, rotating, zooming, panning, and so on), as well as standard motion controls (play-forward, play-back, pause, slow-motion, and so on) on live moving objects. Such spatio-temporally interactive environments can be beneficial in many application areas, including medical or educational applications, especially when it is necessary to study an object, for example a body organ, in action. It can be very attractive in 3D computer games and graphics applications, as well.

In this chapter, assuming that the spatio-temporal sampled description of the object is available or can be generated, we focus on establishing an environment

for interactive, multi-resolution rendering of the object over space and time. This extension is achieved through deploying the MSTree structure introduced in the previous chapter to the case of spatio-temporal samples of moving 3D objects. We also introduce an improved hierarchical quantization scheme based on a hierarchy of hypercubes so that the position and time index of the samples at each level are simultaneously encoded in a compact representation. The implicit, balanced representation, combined with dense hierarchical encoding, results in a compact representation of the moving model for a fairly long sequence of 3D frames where each frame is composed of several thousands of 3D samples. We can also achieve interactive frame rates and quality rendering of such large models on commodity desktop or mobile systems.

8.1 Using MSTree in Balanced, Hierarchical Representation of Animated 3D Objects

Interpreting samples' position and their properties, namely intensity/color, normal, and so on, as multidimensional tuples within a hyperspace suggests the possibility of extending the techniques of hierarchical organization of 3-dimensional points (see Section 6.1) to multidimensional data. Such a general scheme has the advantage of encapsulating all sample properties together and is expected to result in a dense representation of the object. However, in practice, due to little dependency between geometrical properties of samples, namely the position of samples, and other sample properties, namely color, normal, illumination, and so on, these multi-dimensional tuples form a very sparse set within the high-dimensional hyperspace. This makes it difficult to apply efficient clustering/dividing techniques on them during the multi-dimensional break-down process. On the contrary, there is a high correlation between sample properties, geometrical or visual, over consecutive time intervals (frames) which can be used in encoding sampled representation of the moving objects. Specifically, the sample's position and its time index can be efficiently encoded together within the 4-dimensional spatio-temporal space with little or no effect on visual quality of the moving object. Nevertheless, such a generalization often introduces additional challenges to both the preprocessing and the rendering phases. For example, some structures such as QSplat are optimized for a 2-manifold 3D surface (a maximum of 4 children are allowed for each internal node), and some others such as the compact representation of Botsch *et. al.* [18]

are mostly efficient for three dimensions or fewer. Moreover, we face the problem of mapping 4D samples to the 3D/2D space during the rendering. In this regard, the representation should efficiently support some appropriate time-culling techniques to realize smooth object movements across consecutive time frames.

MSTree is a simple and flexible structure that can be deployed to address these challenges and provide a compact and efficient representation for interactive rendering of sampled animated 3D objects. Here we explain how the *bi-partitioning* and *combining* steps of Algorithm 1 in Chapter 7 can be customized to successfully deploy the MSTree structure to the case of these 4D spatio-temporal samples.

8.1.1 Embedding the Time Component within the Tree-Structure

Here we assume that samples are originally in form of (x, y, z, t) where (x, y, z) is the spatial position of the sample, and t is its time index or frame number, and that they are accompanied with a normal and some optional attributes such as color, material and so on. Moreover, we assume that each sample has a radius r which is calculated based on its spatial distance to the neighboring samples in the same time frame. The radius is conservatively calculated to be large enough so that the surface of the object is appropriately covered by samples in each frame.

The bi-partitioning scheme we use here, Algorithm 6, is a simple median-based approach [44]. As explained in Chapter 7, for the case of 3D spatial data such a simple scheme may lead to some serious rendering artifacts, due to inappropriate spatial classification of the points that are not close enough to each other. In the case of 4D spatio-temporal samples, and assuming there are small object movements between the consecutive frames, the spatial density of the whole sample set increases, resulting in a reduced probability of inappropriate classification. Never-

Algorithm 6: Median-based bi-partitioning of 4D spatio-temporal samples.

$(P_1, P_2) =$ ***bi-partition4D***(P)
Input: Set of samples P
Output: Partitions P_1 and P_2

1 $\{d_x, d_y, d_z, d_t\} =$ bounding hyper-box of P;
2 $d = \max\{d_x, d_y, d_z, d_t\}$; `// the longest dimension`
3 $m =$ median of samples on longest dimension d;
4 $P_1 = \{p \in P \| p(d) \leq m\}$;
5 $P_2 = P - P_1$;
6 **return** (P_1, P_2);

theless, the simple median-based approach may still result in some incorrect classifications which are treated using a two-level relative quantization (see Section 8.2). These makes Algorithm 6 good enough for the purpose of our application; however, in general, this algorithm could be replaced with other, possibly more sophisticated clustering techniques.

The 3D (moving) objects are usually sampled at different densities and rates. For this reason, we initially adjust the time index t of the samples to equalize the effect of the spatial and temporal components in the partitioning process. This is done by multiplying the time index t of samples by a nominal time interval or time span value t_s. This nominal value is calculated proportional to the spatial bonding box of all samples of all frames together and the spatial sampling density of the object surface. The *bi-partition4D* algorithm uses the adjusted time indices of the samples in calculating the bounding hyper-box of the input samples. The set of input samples P is divided into two parts P_1 and P_2 along the longest dimension of the bounding hyper-box. Assuming that the median is the average of the two middle numbers for an even number of samples, Algorithm 6 fulfills the assumption that P_1 will either be of the same size as or will get only one more sample than P_2. This simple bi-partitioning scheme implicitly incorporates the time index or frame number of samples into tree structure whenever samples are divided along the (adjusted) time dimension.

8.1.2 Calculating the Internal Nodes of the Tree-Structure

Procedure *Combine* in Algorithm 1 builds the internal nodes in upper levels (lower resolutions) of the tree. Our implementation of *Combine* calculates the parent spatio-temporal position $C_p(x_p, y_p, y_p, t_p)$ as the center of the tightest hypercube containing the hypercubes of all of its children $C_i(x_i, y_i, z_i, t_i)$. Figure 8.1 shows a 2D illustration of these calculations. Assuming d_i is the dimension of the hypercube of child C_i (for the tree leaves d_i is simply the dimension of a cube with the same diameter as the sample diameter $2r$), the lower bound and the upper bound of

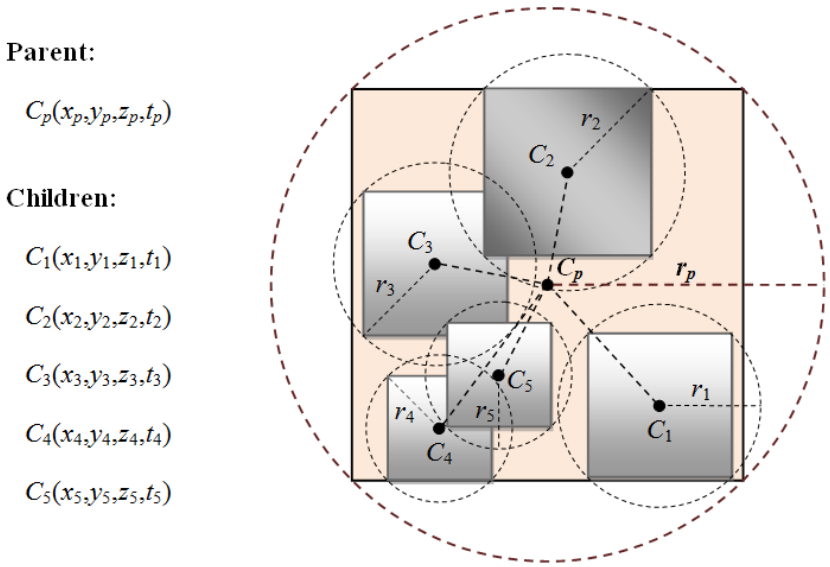


Figure 8.1: Creating parent nodes from lower-level children.

the occupied space by all children are calculated as:

$$\begin{aligned} x_{lb} &= \min_i(x_i - d_i/2), & x_{ub} &= \max_i(x_i + d_i/2) \\ y_{lb} &= \min_i(y_i - d_i/2), & y_{ub} &= \max_i(y_i + d_i/2) \\ z_{lb} &= \min_i(z_i - d_i/2), & z_{ub} &= \max_i(z_i + d_i/2) \\ t_{lb} &= \min_i(t_i - d_i/2), & t_{ub} &= \max_i(t_i + d_i/2) \end{aligned} \tag{8.1}$$

From the above equation the center of the parent node is calculated as:

$$\begin{aligned} x_p &= \frac{(x_{ub} + x_{lb})}{2}, & y_p &= \frac{(y_{ub} + y_{lb})}{2} \\ z_p &= \frac{(z_{ub} + z_{lb})}{2}, & t_p &= \frac{(t_{ub} + t_{lb})}{2} \end{aligned} \tag{8.2}$$

and the parent hypercube dimension d_p is calculated as:

$$d_p = \max(x_{ub} - x_{lb}, y_{ub} - y_{lb}, z_{ub} - z_{lb}, t_{ub} - t_{lb}) \tag{8.3}$$

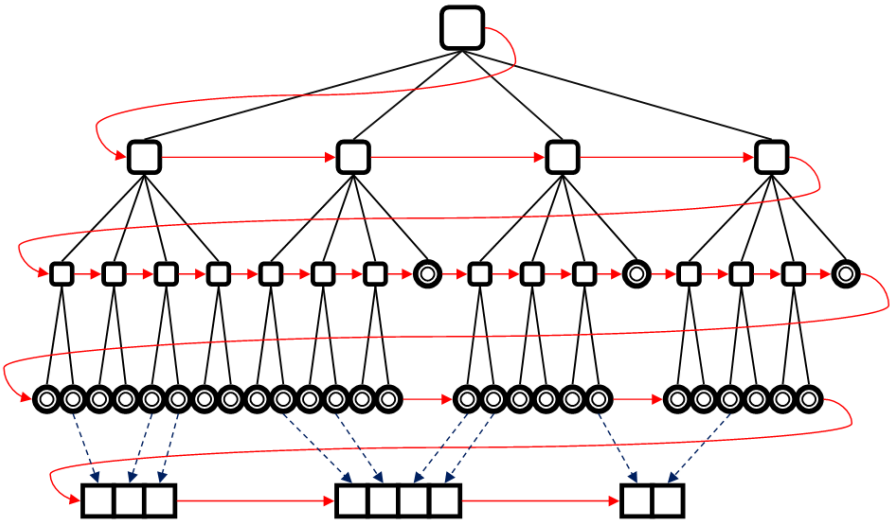
Finally, the parent radius r_p is computed as the radius of the circumscribed hypersphere of its hypercube. In other words, r_p is simply half of the diameter of a hypercube of dimension d_p. The parent node, which is obtained through Equations 8.2 and 8.3, not only spatially contains all of its children but also covers their time

span. This helps to appropriately filter out those samples that are not crossed by the desired (time) frame during the rendering phase (see Section 8.3). Of course, the parent radius may be spatially overestimated if the time frames of its children are not close enough to each other. However, successive breakdowns across the time dimension rapidly dissolves such spatially oversized estimations on finer resolution levels. The bounding boxes (spheres) that are calculated using the above equations are often tighter than those that are obtained by the averaging method used in QSplat [85] to build the bounding spheres. Thus, this method results in more accurate hierarchical quantization for the same quantization bit rate (see Section 8.5 for some comparative results).

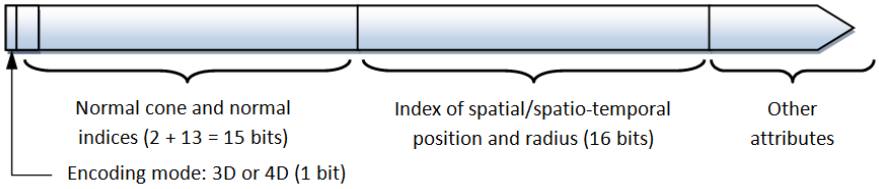
8.2 Hierarchy Quantization and Encoding of Spatio-Temporal Data

Extending the 3D hierarchical encoding [85] to 4D space, here we explain how the children's spatio-temporal position and dimension (radius) are quantized relative to those of their parent. Recall from the previous section that the hypercubes of children are completely contained in the hypercube of their parent. Assuming $(\delta x, \delta y, \delta z, \delta t)$ are the components of the distance vector $\overrightarrow{C_p C_i}$, all of these components as well as the child dimension d_c are smaller than or equal to the parent hypercube dimension d_p. Hence, all child components can be expressed as a fraction of the parent dimension where often a subset of different combinations of these components are valid for a given quantization precision. To get the most compression out of this, the valid combinations are precalculated, indexed, and stored in a look-up table. Then, the quantization task is reduced to finding the index of a valid combination that best resembles the relative position and dimension of a child with respect to those of its parent. For simplicity we call this the STPR (spatio-temporal position and radius/dimension) index. Again, note that quantizing the children's components relative to the parent dimension (instead of parent radius [85]) increases the quantization accuracy. We also found that this method is more numerically stable during our implementation and experiments.

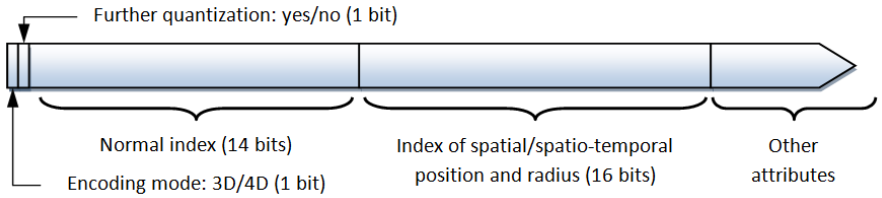
Considering that the balanced MSTree can be represented and stored in an implicit form, there is no need to store any information related to the tree structure and the whole memory space dedicated to a node can be used to store sample data. Here, as depicted in Figure 8.2, we have considered a size of at least 4 bytes or



(a) Order of storing tree on a disk



(b) Internal node layout



(c) Leaf node layout

Figure 8.2: Storing order and node layout of multi-section tree.

32 bits for each node. 16 bits of these are used to store the STPR index. With 16 bits, excluding the invalid combinations, we are able to quantize the ratio of the children's components to the parent's dimension to 10 values ranging from $1/10$ to $10/10 = 1$. To obtain further quantization accuracy, we switch to 3D hierarchical quantization whenever the whole spatial 3D sphere of the parent node falls within the nominal time interval t_s. In other words, we switch to 3D hierarchical quantization whenever the time span of the parent node $P(x_p, y_p, z_p, t_p)$ does not cross two or more time intervals (see Figure 8.3). Switching to 3D hierarchical encoding, 16 bits are good enough to quantize the spatial position and dimension of a children relative to its parent dimension to 22 values ranging from $1/22$ to $22/22 = 1$.

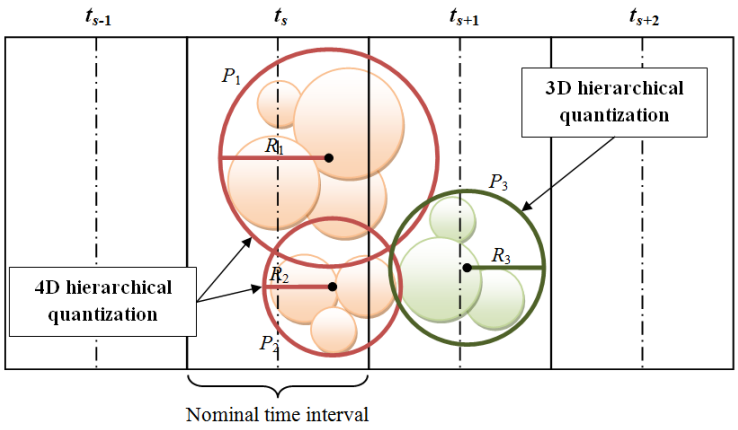


Figure 8.3: Switching between 3D and 4D relative, hierarchical quantization.

This considerably increases the accuracy of representation, which in turn results in improved, quality rendering of the object surface at finer resolutions. As shown in Figures 8.2b and 8.2c, one bit is used to show if hierarchical encoding is in 3D or 4D.

Normals are quantized based on repeated tessellation of a unit sphere into spherical triangles [18]. For the internal nodes 13 bits are devoted to normal index. Starting with a regular spherical octahedron and after 6 tessellation levels we will have $2 * 4^6 = 2^{13}$ spherical triangles, which can be encoded in 13 bits. The other 2 bits are used to quantize the width of the normal cone, which is used for back-face culling, to four values $1/16$, $4/16$, $9/16$, and $16/16 = 1$. According to [85], on a typical 3D data set, this level of quantization is sufficient to discard over 90 percent of the nodes that would be culled using exact normal cone widths. However, at the moment, we have not tested whether the same claim can be made for the case of 4D spatio-temporal data sets. For the leaf nodes, normals are quantized using 14 bits. This time we start tessellation with a spherical tetrahedron and proceed to 7 levels which gives us $4^7 = 2^{14}$ spherical triangles.

Depending on the data set, tree nodes could have some optional attributes such as color, material properties, and so on. These attributes can be quantized accordingly and preferably in multiples of byte for easier and more efficient decoding.

Recalling the challenges of appropriate equi-partitioning (see Section 7.2 and also [10]), some samples (usually very few) may not be appropriately clustered, causing significant quantization error even with the improved quantization scheme. This quantization error shows itself in the form of rendering artifacts in the rendering stage. To address this problem, here we test the leaf nodes for the precision of the quantized samples with respect to the original ones. We calculate the Eu-

clidean distance between the original and quantized sample, and use the ratio of this distance to the sample radius to estimate the quantization accuracy. In case of significant difference, which is decided using a threshold value, we add a second level of quantization assuring that the sample's position and radius are accurate enough when restored and rendered. The STPR indices of these few doubly-quantized leaves are stored in a series of lists right after the tree (see Figure 8.2a). For these samples, we need two additional bytes (16 bits) to store this extra information.

The whole balanced hierarchy is written into the disk starting from the root and following a breadth-first order. As already mentioned, we do not need to store any explicit addressing or label the nodes as internal node or leaf. The tree branching factor and the total number of samples are enough to determine the address of children and also distinguishing the leaves from the internal nodes (see Section 8.3 for details). This results in a compact representation of the spatio-temporal data sets with memory usage of about 5.5 bytes per 4D sample with increased quantization accuracy compared to other representations. The amount of the memory usage may vary depending on the branching factor, number of samples and length of sequence.

8.3 Interactive, Animated Rendering of Restored 3D Model

Algorithm 7 shows the main steps in multi-resolution rendering of spatio-temporal samples. Starting from the root, and following an adaptive depth-first tree traversal method, the algorithm recursively decodes the lower-level children relative to their parent. Figure 8.4 shows a pictorial description of this process. For each child C of parent P, B_C, L_C, S_C, and O_C are calculated as defined and explained in Section 7.1.2. Having the child offset, O_C, the algorithm can extract the child information including the STPR, normal, and normal cone indices. Depending on the type of relative encoding, 3D or 4D, which can be determined by examining the leading bit of each node, the algorithm uses the corresponding 3D or 4D look-up table to find the spatial or spatio-temporal position and radius of C relative to those of P. In case of $3D$ encoding, the temporal position of C is set as the temporal position of its parent P (Steps 4 to 8 in Algorithm 7). Similarly, the normal index is used to find the sample normal from the normals' look-up table(s) and the normal cone index is used to calculate the width of the normal cone.

Algorithm 7 applies different conventional culling techniques such as frustum culling [90] and back-face culling [67] to discard most of those samples that are not visible to the user [85]. In addition, we have the special time culling technique (Steps 9 and 10) which is, in fact, the main step that enables animated rendering of a dynamic 3D object over time. For time culling, we examine whether the time

Algorithm 7: *Multi-section* tree traversal and progressive rendering of point-based animated 3D objects.

TraverseMSTree4D$(P(x_p, y_p, z_p, t_p, r_p), N_P, L_P, S_P, l)$
Input: $P(x_p, y_p, z_p, t_p, r_p)$: Parent node,
B_P: Number of leaves beneath P,
L_P: Number of leaves in the left hand side of P,
S_P: Number of siblings in the left hand side of P,
l: Current level

```
1  Determine number of children of P;
2  for (each child C of P) do
3      Determine B_C, L_C, S_C, and O_C;
       // Determine spatio-temporal position and radius of C
4      if encoding is 4D then
5          Determine x_c, y_c, z_c, t_c, and r_c of C;
6      else
7          Determine x_c, y_c, z_c, and r_c of C;
8          Set t_c as of the parent node P;
       // Check for different culling types
9      if C does not cross the current time span then
10         Continue;                                   // time culling
11     if C is out of the frustum then
12         Continue;                                   // frustum culling
13     if C does not face the viewer then
14         Continue;                                   // back-face culling
       // check if we recurse
15     if C is a leaf then
16         if time of C is within the current time interval then
17             Draw C;
18     else if C is too small OR is smaller than current splat size then
19         Draw C;
20     else
21         TraverseMSTree4D(C, B_C, L_C, S_C, l + 1);
```

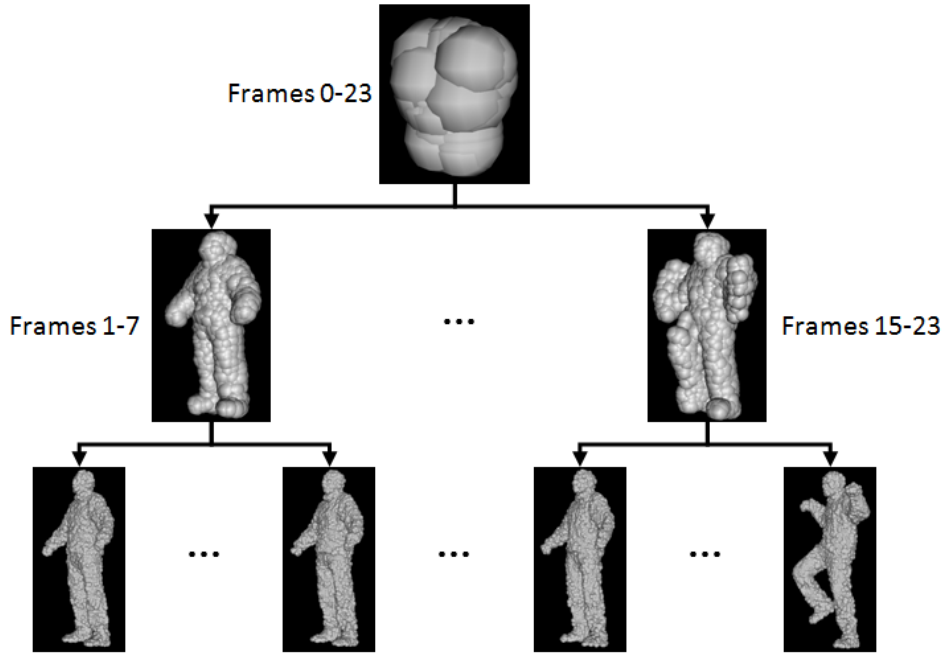


Figure 8.4: Hierarchical rendering of a dynamic 3D model. The adaptive deepening threshold regulates both spatial and temporal details.

span of the given node has any overlap with the current animation time interval $t_a \pm {}^{t_s}/_{2.0}$. The node (and all its underneath subtree) is culled if it satisfies the following condition:

$$(t_a + \frac{t_s}{2.0} < t_c - r_c) \text{ OR } (t_c + r_c < t_a - \frac{t_s}{2.0f}) \tag{8.4}$$

After various culling steps, the algorithm checks whether it is necessary to recurse or whether the node should be drawn immediately. A splat is drawn if the node is either a leaf or the projected size of the splat on display screen is so small that no more quality is gained with looking further into the details of the lower levels. Checking for a leaf (Step 15) is as simple as testing whether or not B_C is equal to 1. At the leaf level another time-related refinement is performed based on the time position t_c of the sample alone, and regardless of the length of the sample radius r_c. This filters out the overlapping samples from neighboring frames which according to our experience leads to considerable improvement in the rendering of a single frame and also generates a smooth, noiseless transition among the consecutive frames. If the node is not a leaf ($B_C > 1$), then the approximate size of the corresponding splat on the display screen is checked; and if it is too small (*i.e.* smaller than the pixel size), or is smaller than an adaptive threshold, the

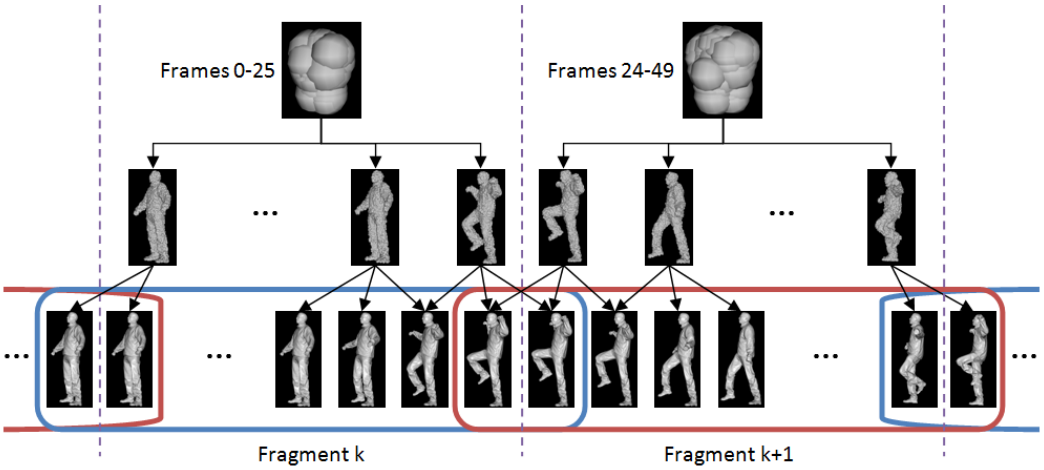


Figure 8.5: Streaming and rendering large and long dynamic sequences.

algorithm stops further deepening and draws the sample right away. The threshold is adapted proportional to the time needed for processing and drawing individual samples to achieve the desired frame rate. In fact, even though a depth-first tree traversal scheme is applied, the level of deepening is decided by different environmental conditions, most of them succinctly reflected by the value of the adaptive threshold. The algorithm goes into further details (Step 21) only if none of all the above-mentioned conditions are satisfied. Note that the threshold value concurrently regulates both spatial and temporal details (see Figure 8.4 again) which makes this structure very efficient in streaming and rendering large dynamic models.

The animation time t_a is a parameter that is either automatically incremented by the application on a regular time interval or controlled by a user. The first mode is called *playing mode* and the latter one is called *navigation mode*. A user is able to switch between these two modes by pushing a play/pause button similar to a conventional video player application. In navigation mode the user is able to increase or decrease the current frame number, or jump to a specific frame including the first and the last frame of the sequence. The user is also able to do all the conventional spatial navigations, such as panning, zooming, rotating, spinning, and so forth, either in playing or in navigation mode.

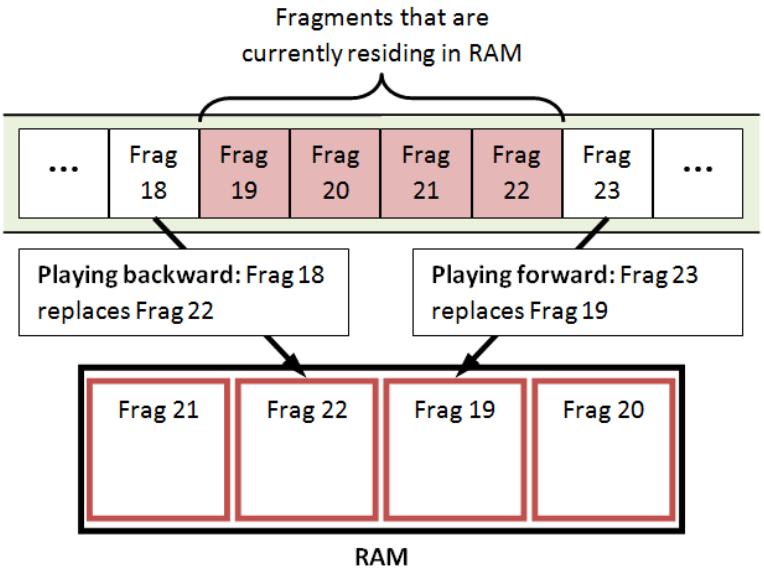
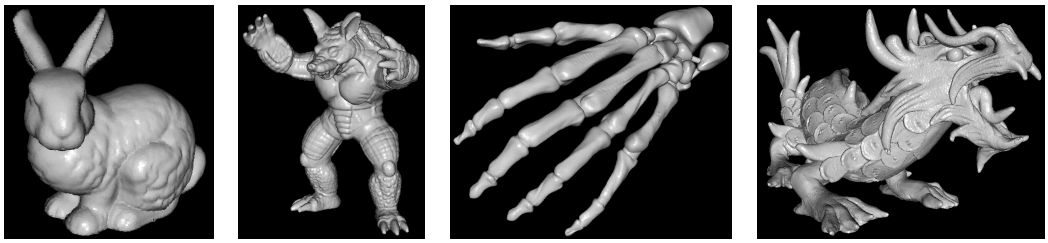


Figure 8.6: Circular management of device RAM for streaming and rendering large sequences.

8.4 An Architecture for Streaming Large Dynamic Point Clouds

For real time streaming of dynamic point clouds we need to make efficient updates to the data stored on RAM based on the current viewing location. Real time visualization coupled with real time streaming can be enabled by storing a list of consecutive data blocks or fragments in memory that contains $(t_c - T)$ to $(t_c + T)$, where t_c is the current frame index and T is determined based on the capacity of the device RAM. As depicted in Figure 8.5, each of these fragments contains a balanced tree built on a set of successive frames, where adjacent fragments may have one or more boundary frames in common. These overlapping frames aid in the smooth transition between successive fragments in playing mode.

The device RAM itself is managed in a "circular" way (see Figure 8.6). When the sequence is playing forward, the next right-hand-side fragment is read into memory and replaces the fragment with the smallest frame number. Conversely, when the sequence is playing backward the next left-hand-side fragment is read and substituted for the fragment with the largest frame index. Note that because the data blocks are spatio-temporally coded, the actual information loaded into memory does not need to be the entire body of information as in the case of loading a sequence of video frames. In fact, as already explained, depending on the amount



(a) (b) (c) (d)

Figure 8.7: The 3D models that are used in calculating statistics in Table 8.1: (a) Stanford Bunny, (b) Armadillo, (c) Hand, and (d) Chinese Dragon.

Table 8.1: Comparing accuracy of cube-based hierarchical quantization (our method) versus the sphere-based method used in QSplat.

		3D Model			
		Bunny	Armadillo	Hand	Dragon
Number of Samples		34834	172974	327323	3609600
Mean Sampling Distance		0.00147	0.52391	0.01232	0.11552
Spherical Encoding	Mean	0.00022	0.07084	0.00168	0.01552
	Std Dev	0.00011	0.03330	0.00078	0.00772
	Max	0.00587	1.82827	0.02604	1.21989
Cubic Encoding	Mean	0.00014	0.04794	0.00114	0.01102
	Std Dev	0.00005	0.01777	0.00043	0.00436
	Max	0.00176	0.85422	0.01423	0.65409
Mean Error Ratio (%)		60	68	68	71

of the device RAM and/or the channel bandwidth, the level of spatial or temporal details can be controlled using the adaptive threshold. Therefore, the user will be able to get a general sense of the motion of the object in real time even with limited resources, but should wait for transmission and loading of all data if he/she wants to see all details.

8.5 Experimental Results on Animated Point-based Rendering

Table 8.1 shows some quantization error statistics calculated based on two different quantization schemes: the sphere-based hierarchical quantization used in QSplat, and the cube-based hierarchical method we have used in our implementation. The statistics are calculated for four different still 3D models shown in Figure 8.7. Both

Table 8.2: Compactness of the representation for some animated 3D meshes of different length and different number of samples per frame.

Model Name	Number of Frames	Total Number of Samples	Number of Samples per Frame	File Size (Bytes)	Number of Bits per 4D Sample
Bee	72	7,551,216	104,878	41,091,284	43.53
Crane	48	7,680,096	160,002	41,301,576	43.02
March	250	2,500,500	10,002	14,573,072	46.62
Samba	175	6,978,650	39878	38,811,660	44.49
				Average	44.42

methods are applied to the same quadtree structure and the same number of bits (13 bits) are used for relative quantization of the spatial position. The statistics (mean, standard deviation, and maximum error) are calculated from the differences (in terms of the Euclidean distance) in the original (non-quantized) points position, and their corresponding position which is reconstructed from the quantized representation. It can be seen that for all four models of different topological complexity, the cubic method results in a more accurate representation through an increase of some 30 to 40 percent in quantization precision (see the last row of Table 8.1). More importantly, during the experiments we noticed that the cubic method is numerically more stable, especially in the boundary conditions and for the case of 4D samples.

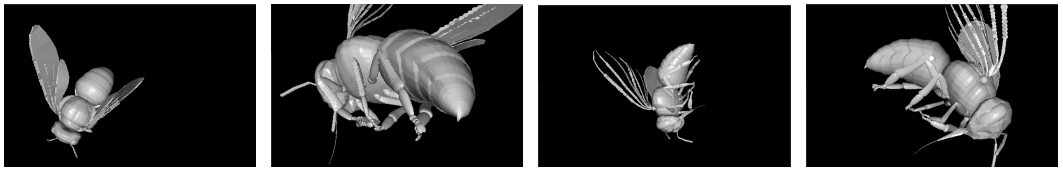
Figures 8.8 and 8.9 show a series of rendering results for animated 3D objects composed of spatio-temporal 4D samples in *navigation* mode and *playing* mode, respectively. The user switches between these two modes by simply clicking on a play/pause button. In the navigation mode, when the playing is paused, the user is able to move the playing head to the previous, next, first, last or any other specific frame. In addition, the user is able to perform all of the spatial interactions such as rotating, zooming, panning, and spinning on the single frame which is navigated to or paused on. Figure 8.8 shows some of these capabilities for a mesh sequence called Crane. This sequence is composed of 48 frames with, on the average, 160,002 samples in each frame and 7,680,096 samples in total (see also Table 8.2).

In playing mode, where the frames of the sequence are rendered one-by-one based on a regular time interval, the user is able to apply the spatial interactions at the same time as the object moves. This enables interactively watching the desired parts of the sequence (the moving object) from closer distances, different perspectives, and viewing fields. Figure 8.9 shows some snapshots of applying some spatial

(a) First frame (b) Frame 26 (c) Frame 27

(d) Frame 27, Rotating (e) Frame 27, Zooming (f) Frame 28 (g) Last frame

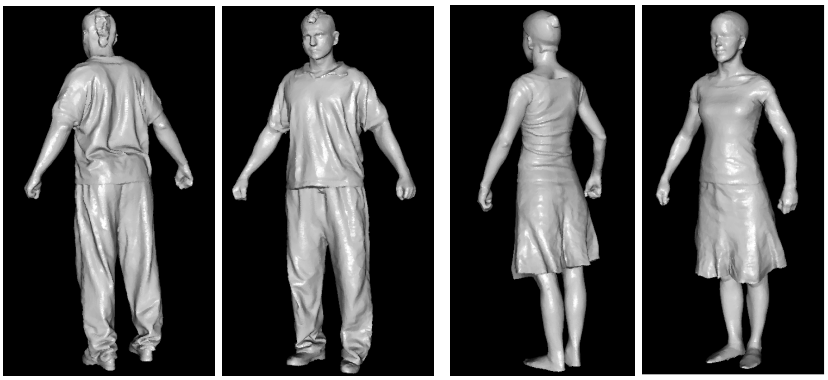
Figure 8.8: Rendering in navigation (or pause) mode. Navigation mode provides the possibility of going to the next, previous, first, or last frame, and also allows for spatial interaction (rotating, panning, zooming, and so on) with the frame that user is navigated to/paused on.



(a) (b) (c) (d)

Figure 8.9: Rendering in playing mode with possibility of simultaneous application of all spatial interactions such as panning, zooming, rotating, and spinning on the moving object.

navigations on the Bee sequence in playing mode. The Bee sequence is composed of 72 frames, with 104,878 samples in each frame. As perceived, the user is able to watch the flying bee from different angles and distances whereas in the default position and orientation the bee is visible to the user only from the upper side throughout



(a) March (b) Samba

Figure 8.10: Front and back face of the first frame of (a) March and (b) Samba sequences.

the sequence.

Table 8.2 summarizes some statistics related to the size and compactness of four sequences of different length and different number of samples per frame. These include the Bee, Crane, March, and Samba models. The front and back side of the first frame of March and Samba are shown in Figure 8.10. The file size (or more accurately the number of bits per 4D sample) depends on different parameters including the branching factor of MSTree and also the number of frames and number of samples per frame. The values in Table 8.2 are calculated based on using a MSTree structure of branching factor 8. For this representation, on the average, we need about 44 bits per sample to encode the spatio-temporal position, radius, and normal of samples at all resolutions.

We are able to render all the sequences in Table 8.2 in interactive frame rates. Specifically, we are able to play the Crane sequence, which has the highest number of samples per frame among these sets, at 20 to 30 frames per second (depending on the rendering primitive) on commodity laptops and desktop machines. It should be clarified that the adaptive, multi-resolution rendering technique used in Algorithm 7 allows interactive frame rates to be achieved for models of higher number of samples per frame and on less powerful machines but at lower resolutions over space and time.

Part III

Interactive Manipulation of Sampled Object Representations

Chapter 9

3D Interaction and Manipulation

Along with the advances in stereoscopic display technologies it is also necessary to evolve the current 2D/3D interaction techniques and devices to work with these new attractive virtual 3D environments. Among different interactions, the ability to point to the targets (objects or other GUI components) using devices such as a mouse is probably the most common and appealing method of interaction. This ability is especially necessary and useful for working in a graphics environment [33]. For this reason, providing the possibility of pointing to any arbitrary voxel of the 3D space is one of the primary requirements of any application functioning in 3D space. In this part we study 3D interaction and manipulation within a virtual stereoscopic 3D space. In this regard, we begin this chapter by briefly reviewing some basic concepts and techniques of 3D interaction and manipulation. Then in the next chapter we study a stereo-based 3D cursor, and its capabilities and weaknesses as a generic pointing tool within a stereoscopic 3D space. We also discuss the pointing accuracy of the stereo-based 3D cursor with regard to the optimal sampling model proposed and discussed in Part I.

9.1 Three-Dimensional Pointing and Interaction Techniques

Several efforts have been made over the years to simplify interaction with 3D application environments. On one hand, benefiting from different mechanical, electromagnetic, optical, acoustic, and inertial sensors and technologies, several kinds of 3D input devices have been introduced to facilitate working with 3D models and applications [47, 40]. On the other hand, several task-specific or general-purpose

interaction techniques have been developed [33] which in combination with the enhanced 3D input devices and technologies provide an easier, more natural way of handling tasks within a 3D environment. In general, these interactions can be classified into three different task domains [40, 48]:

- Object selection and manipulation
- Viewpoint manipulation (navigation and travel), and
- Application (system) control

Pointing to the targets (objects or other application components) may be considered one of the most widely used interaction techniques in GUIs; it is a prerequisite for performing many tasks in each of the above-mentioned task domains. A pointing task is usually achieved by manipulation of a 2D/3D cursor position and/or orientation using an appropriate input device.

A chronological survey of several 3D pointing techniques and 3D cursor concepts is presented in [33]. These include Skitters and Jacks (1987), Ray casting (1994), Spotlight (1994), Virtual hand (1995) and many other older or recent techniques. According to the survey, all of these methods are based mainly on a virtual hand, ray, or spot-light technique. Following this classification, a more formal definition for a 3D cursor is presented. The definition is based on two assumptions and a set of requirements and constraints that together have to be satisfied by a typical 3D cursor as follows.

- **Assumptions:**
  - The pointing device has 6 or more (at least 3 translational and 3 rotational) degrees of freedom (DOF).
  - The selectable target has to be visible to the user.
- **Requirements and constraints:**
  - Visual presentation - the 3D cursor has a graphical presentation which makes its position and orientation observable to the user.
  - Behavior - the movements of the 3D cursor are controllable by the user using an appropriate input device.
  - Constraints - the 3D cursor reaches all the positions on the (3D) graphical display and is able to touch only one target at a given instant.

Based on this definition, two main types of 3D cursors are considered for 3D UIs: the 3D point cursor and the 3D line cursor. These two main types satisfy all above-mentioned requirements, and all aforementioned 3D pointing techniques can be derived from them [33]. In fact, the other 3D pointing techniques may be considered to be the result of exploiting some *virtual enhancements* to improve the performance of the 3D pointing (or the 3D target acquisition time). Here, a virtual enhancement means changing the values of one or more parameters effective in the 3D target acquisition time according to the 3D Fitts' law. The 3D Fitts' law formulates the target acquisition time (or the average movement time) to select a target. The law states that the target acquisition time or the *average movement time* (MT) to select a target depends on the *distance to move* (A), the *target size* (W: width, H: height, and D: depth of the target), and also the *viewing angle* (θ) at which the target is seen by the user through the following equation:

$$MT \approx 56 + 508 \log_2 \left(\sqrt{f_W(\theta) \left(\frac{A}{W}\right)^2 + \frac{1}{92} \left(\frac{A}{H}\right)^2 + f_D(\theta) \left(\frac{A}{D}\right)^2} + 1 \right) \tag{9.1}$$

where $f_W(0^\circ) = 0.211$, $f_W(90^\circ) = 0.717$, $f_W(45^\circ) = 0.242$, $f_D(0^\circ) = 0.194$, $f_D(90^\circ) = 0.312$, and $f_D(45^\circ) = 0.147$. In this regard, reducing the movement distance A, increasing the target size (or equivalently the cursor size), or a combination of these changes can be considered as examples of such virtual enhancements.

9.2 3D Pointing and Interaction within the Stereoscopic 3D space

The review in [33] does not refer to the interaction techniques and 3D pointing in a stereoscopic 3D environment; however, the same concepts and principles can be applied to the case of stereo. 3D pointing in stereo space can be achieved through using a (multiview) stereo cursor. Such a cursor can be created by providing two or more views of an ordinary 2D cursor at specific distances or disparities. The cursor depths can be controlled by adjusting the amount of the disparity. This enables the user to point to any arbitrary 3D location inside the virtual 3D space projected by a stereoscopic 3D display. This simple technique has been known for years and has already been applied in some stereoscopic 3D visualization and manipulation applications and GUIs. For example, OrthoEngine 3D Stereo offers a 3D stereo

cursor among its advanced tools for the 3D viewing and manipulation of aerial pictures or satellite imagery data [2]. BioMedCache as an application for molecular design also offers stereoscopic 3D visualization and 3D stereo cursor [1]. In [86], authors present their success in modifying the functionality of X Window System with the purpose of constructing generic tools for displaying 3D-stereoscopic content. In this context, they refer to the implementation of a 3D pointer through the creation of a shadow pointer which follows the motion of the real pointer in both fields of the stereo window. In their implementation the depth of the 3D cursor is controlled by automatic adjustment of the disparity of the shadow pointer with respect to the real one. More recently, the prototype of a simultaneous 2D/3D GUI for (auto)stereoscopic displays is introduced that includes implementation of a 3D stereo cursor. The stereo cursor is enabled upon the entry of the mouse pointer into the 3D GUI area [97]. Here again the disparity is automatically adjusted to keep the virtual 3D cursor in touch with the surface of the 3D objects and 3D GUI components.

In the aforementioned efforts, the stereo cursor has usually been implemented as a by-product of the stereoscopic 3D visualization systems, and less attention has been paid to its capabilities as a generic replacement of the 2D cursor in the stereoscopic 3D space. In the next chapter, Chapter 10, we study the problem of 3D pointing in a stereoscopic 3D environment with a focus on different aspects of the stereo 3D cursor itself [8, 9]. Our stereo cursor implementation essentially follows the same principles applied by others to form the 3D cursor. However, it supports two disparity adjustment modes (manual and automatic) which in fact enables us to show that such a simple technique satisfies all of the requirements of an abstract 3D cursor. Furthermore, we show how the scene stereo content may be used to virtually enhance the performance of the technique according to the 3D Fitts' law. We have also conducted user tests for comparing the effectuality of 3D cursor to the use of an ordinary mouse cursor on a conventional 2D screen.

Chapter 10

Stereo 3D Cursor: A Generic Method of Interaction within a Stereoscopic 3D Space

The stereo-based 3D cursor is a 3D pointing technique that has usually been implemented as a by-product of stereoscopic 3D visualization (and/or 3D object manipulation) systems or applications with less attention to the capabilities of the technique as a generic replacement of a 2D cursor in stereoscopic 3D space. In this chapter we focus on different aspects of the stereo 3D cursor, hereafter also S3DCursor, and show how it satisfies the main functionality requirements of an abstract 3D cursor. We use a multi-view stereo rendering application to evaluate the capability of the 3D versus 2D cursor in manipulating visualized 3D data, and show how the content of a scene can be used to virtually enhance the simplicity and efficiency of object selection and pointing tasks within a virtual 3D space. We also discuss some issues such as pointing accuracy and the problems caused by occlusion, and possible methods of handling these problems. In general, our evaluations suggest the effectiveness of the technique in terms of several factors including detection accuracy, simplicity of usage, and overall user satisfaction compared to using an ordinary cursor on a conventional 2D screen.

10.1 Stereo Cursor Implementation

The stereo 3D cursor is obtained by providing replicates of a 2D cursor on two (binocular) or more (multiview) perspective views of a scene at a specific disparity.

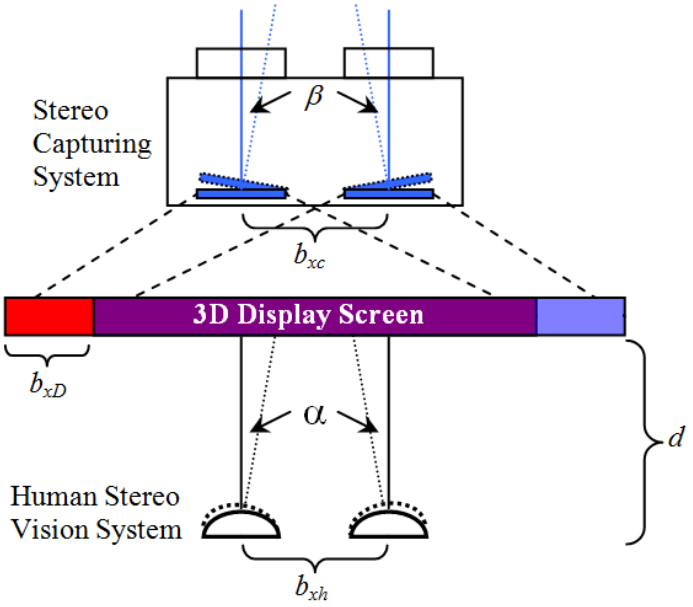


Figure 10.1: Process of capturing, displaying, and watching stereo images.

The cursor depth can be controlled by adjusting the amount of the disparity between these replicates. Hence, changing the cursor disparity enables the user to point to any arbitrary 3D location inside the virtual 3D space projected by a stereoscopic display.

To examine the applicability and usefulness of the idea we needed a stereoscopic 3D environment for interactive 3D multiview rendering and manipulation of the object. For this purpose, we extended the QSplat rendering engine, which uses OpenGL as its graphics library, to support rendering multiple views of a 3D object. The extension, which is called MVSplat, supports the original QSplat model representation [85], POP point-polygonal representation [22], and also some of the balanced representations introduced in Part II. MVSplat enables the user to decide on the number of cameras (views), the distance between the cameras b_{xc}, and the amount of horizontal displacement of the views on the display screen b_{xD} (see Figure 10.1 and also Figure 2.5). The current version assumes parallel configuration and the same baseline for all cameras. This allows a simplified implementation of the cameras' rotation and translation. These movements are applied to a central virtual camera, and then the position and direction of all cameras are set with respect to this virtual camera. The calculations of cursors' disparities are also simplified under parallel geometry. The system also supports a special red-blue rendering mode which offers users the flexibility to use the application on all conventional displays just by wearing simple anaglyph glasses.

The stereo cursor is implemented on the top of the MVSplat. Two different disparity adjustment modes are considered for the cursor: *manual* and *automatic*. In the automatic mode the application uses the 3D content of the scene to automatically adjust the disparity of the 2D cursors (the depth of the stereo cursor) so that the 3D cursor always remains in touch with the surface of the object(s) present in the scene. The automatic mode is more useful in applications such as 3D games, where it is often necessary to target the 3D objects already existing in the scene, and there is usually no need to create new objects. In the manual mode the user is able to change the depth of the 3D cursor by manually adjusting the disparity between the left and right cursors. The manual mode is useful when the user wants to point somewhere other than the visible surface of the 3D object or to adjust the disparity estimated by a stereo matching algorithm. The latter case in particular can be used as a cost-effective, accurate method for extracting ground-truth data from stereo image pairs.

Our implementation does not imply any assumptions about the input pointing device and fully complies with the current functionality of 2D mice. However, special input devices, such as the 3D mice currently available, may facilitate the performance of 3D tasks. Moreover, the disparities are calculated concurrently with the 3D model rendering process; therefore, it does not impose an extensive additional computational load on the application. This issue will be completely resolved by providing a system level support for stereo cursor.

10.1.1 Binocular Implementation

To implement the automatic mode in binocular view, one view, for example the left one, is considered as the reference view. When the left cursor points at a pixel in the left view, having the depth information of the pixel (usually available in the depth buffer) and the camera parameters, the corresponding pixel in the right view (or the position of the right cursor) can be determined using the basic stereo imaging formulations as follows.

Considering the OpenGL default perspective projection [51], which implies a normalization transformation as well, the relationship between the point depth in the virtual camera coordinate Z_c and the depth maintained in the depth buffer Z_w can be stated as:

$$Z_c = \frac{f_c d_{far}}{Z_w(d_{far} - f_c) - f_c} \tag{10.1}$$

where d_{far} is the far clipping plane distance and f_c is the near clipping plane dis-

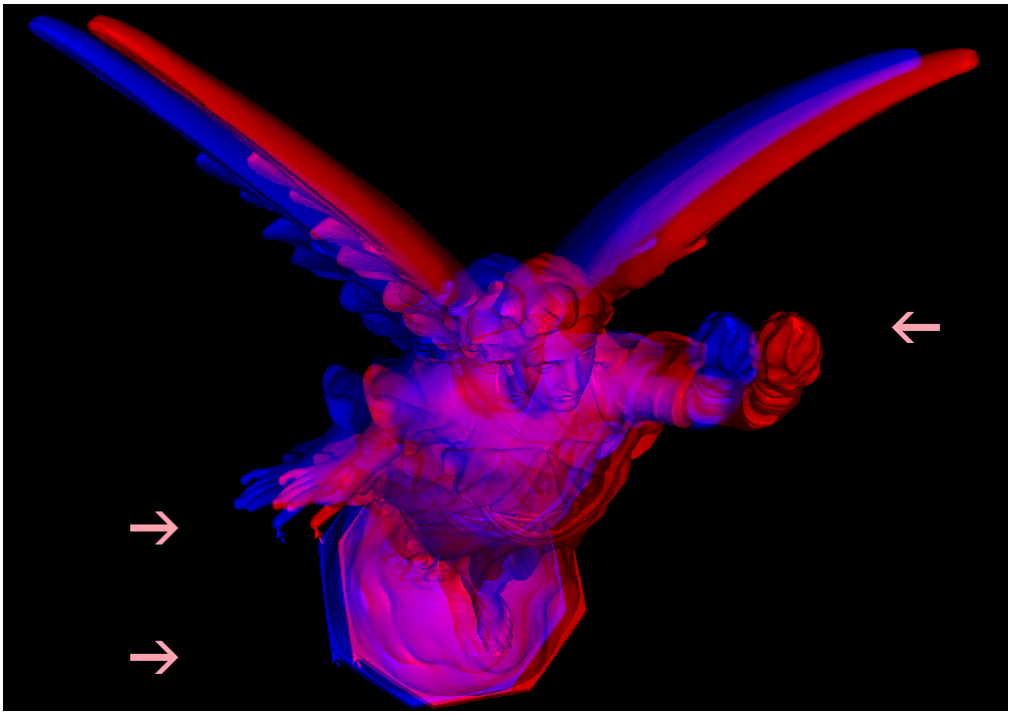


Figure 10.2: Red-blue stereo rendering of the Lucy model with three samples of the stereo mouse cursor presented at different distances (disparities).

tance or camera focal length. On the other hand, from Equation 2.2 Z_c is determined as:

$$Z_c = \frac{f_c b_{xc}}{x_{rc} - x_{lc}} = \frac{f_c b_{xc}}{disp} \tag{10.2}$$

From 10.1 and 10.2 the amount of the disparity is obtained as:

$$disparity = \frac{(Z_w(d_{far} - f_c) - d_{far})b_{xc}}{d_{far}} \tag{10.3}$$

The disparity calculated in Equation 10.3 should be properly scaled and adjusted considering the viewport transformation settings and the amount of the views' displacement b_{xD}.

The automatic disparity adjustment causes the illusion of touching the surface of the real 3D object so that when the user slides the mouse pointer over the display screen, the virtual 3D cursor follows the holes or other irregularities on the 3D object surface. Figure 10.2 shows a red-blue stereo pair of the Lucy model with three instances of the stereo mouse cursor at different disparities which, in fact, form three 3D cursors at three different distances from the viewer (readers may watch this figure in color and using red-blue anaglyph glasses to see the formation

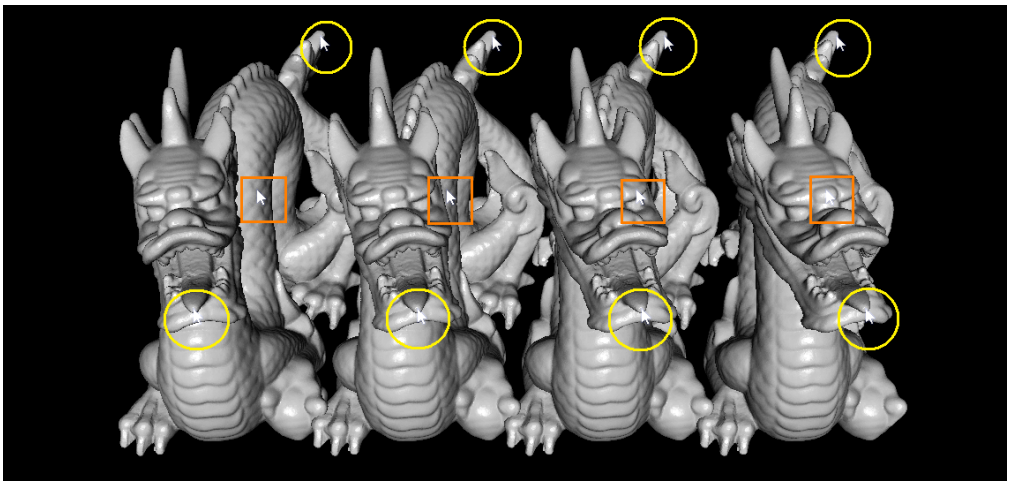


Figure 10.3: Four-view stereo rendering of the Dragon model with three samples of stereo cursors on occluded (squares) and non-occluded (circles) areas.

of these 3D cursors at different distances).

Implementing the automatic disparity adjustment over the image/video stereo pairs implies that an efficient stereo matching algorithm should be incorporated into the system to find the corresponding projections in the left and right views. Contrary to the classic stereo matching algorithms, which requires the correspondence for all pixels to be established, here the correspondence needs to be found only for the pixel located under the current position of the left (or alternatively right) mouse cursor. This assumption may lead to finding more efficient algorithms for real-time applications.

The manual disparity adjustment is simply achieved by allowing the user to change the amount of the cursor disparities using some appropriate input device (keyboard or mouse) keys and/or buttons.

10.1.2 Multiview Implementation

The multiview implementation is particularly useful for autostereoscopic multiview displays. The implementation is essentially similar to the two-view case. Assuming all cameras are parallel and located on the same baseline at equal distances, all corresponding projections of a 3D point on the display screen will be located at the same raster line with the same disparity. As a result, one of the views can again be considered as a reference for disparity calculations, and the other corresponding projection can be determined with respect to the reference view.

Although this implementation is pretty simple, it may cause some problems in

occluded areas. This is because in the autostereoscopic 3D displays the viewer is only able to see two consecutive views of the scene at a time. As a result, as depicted in Figure 10.3, the implementation works fine while the corresponding projections, similar to the instances inside the circles, are visible in all views. The implementation becomes problematic when corresponding projections fall into occluded areas in two or more consecutive views. As an example, the reader may observe the third and the fourth view of the cursors surrounded by squares. When the viewer moves his/her head to watch the third and the fourth views, the 3D position of the 3D cursor reconstructed from these two cursor views is incorrectly estimated. The problem could be fixed if each view served as the reference for the next adjacent view. However, this may lead to the ambiguity in converting the cursor positions to a unique 3D position. A more advanced algorithm might detect the occluded areas and hide or highlight the cursor in corresponding views. Upon receiving such a hint, the user may change the viewpoint or the cursor position to access a specific point from any desired view.

10.1.3 Stereo Cursor as a Generic 3D-Pointing Technique

In general, the stereo cursor can be considered a 3D pointing technique which satisfies all of the assumptions and requirements of an abstract 3D cursor as follows (see also Section 9.1).

- It obviously has three translational DOFs, and three rotational DOFs can be achieved by viewpoint manipulation.
- Although the occlusion problem on autostereoscopic displays may necessitate a further move to grab the target, all visible parts of the scene (all visible targets) are selectable by the user.
- The cursor has a visual presentation that makes the cursor position observable to the user. This visual presentation can be used to present the orientation of the cursor as well.
- The stereo cursor movements are controllable by the user using a conventional mouse; however, more appropriate input devices such as 3D mice may be adapted for efficiency purposes.
- The stereo cursor is able to reach to all positions in the comfortable viewing range of the stereoscopic display through appropriate (manual) adjustment of

the disparity. This is especially important when the user aims to point to an empty space, for example, for the purpose of creating a new object.

- Finally, since in a stereoscopic 3D space the user only sees the surface of the opaque objects, (s)he will be able to touch only one target at a time in automatic mode (a priority mechanism may be applied for transparent or translucent objects). In manual mode appropriate visual hints may be implemented to inform the user of moving the cursor to the physically invisible areas.

Regarding these properties, the stereo cursor can be applied as a general alternative of 2D cursor in stereo space. Some virtual enhancements may be implemented to improve the basic functionality of S3DCursor. In fact, automatic disparity adjustments may already be considered as such an enhancement which virtually reduces the distance to the target (reduces A in Equation 9.1) by "removing the empty space between the cursor and the targets" - the enhancement that according to [33] has not already been tried in other 3D pointing techniques. "Increasing the target size", *i.e.* increasing W, H, or D in Equation 9.1, is another enhancement which is especially useful in handling the accuracy deficiencies of the stereo cursor (see Section 10.1.4). If the application control components are also implemented in 3D, then several enhancements may be applied to the GUI components, especially on menus and on the application window itself. These include the appearance of the pop-up menus at the same depth as that of the 3D cursor, and dynamically managing the position and size of the windows depending on the scene composition and user actions. These types of enhancements also may be considered as virtual reduction of distance to the target.

10.1.4 Stereo Cursor Accuracy

According to the analysis in Section 2.2, the whole stereoscopy system depicted in Figure 10.1 can be considered as a parallel stereo system whose focal length is equal to the display viewing distance, d, the distance between its cameras or its baseline length is equal to the distance between the human eyes, b_{xh}, and the display screen displaced to the left/right has the role of its left/right image plane. In Part I we discussed and showed that for this stereo setup, given a total resolution R, a finer horizontal discretization, e_x, versus vertical discretization, e_y, yields less error in the estimation of the original 3D scene. Applying this result to the stereo cursor, we can say that a finer horizontal resolution not only yields a better 3D

visualization but also more accurate stereo-based 3D pointing, especially across the depth dimension.

Another issue related to the stereo 3D cursor accuracy is its heterogeneous behavior mainly across the depth dimension. This results fundamentally from the intrinsic behavior of the perspective projection combined with the discretized nature of digital images. As illustrated in Figure 2.9, all points located inside the 3D quadrilateral (or voxel) formed by the corresponding pixels are estimated to the same 3D point. These voxels are non-uniformly distributed so that the resolution (especially the depth resolution) decreases with the distance to the stereo cameras (viewer) [71]. As a result, the stereo cursor is not sufficiently accurate when the objects are not close enough to the viewer. Several virtual enhancements may be applied to compensate for this drawback up to some acceptable extent. Zooming across the third dimension (bringing the target to the cursor), enlarging the objects across the depth (changing the target size across the depth *i.e.* changing the D parameter in Equation 9.1), or manipulating the target under a fish-eye implementation (which again may be interpreted as a means of bringing the target to the cursor) are among the possible enhancements. Another possibility is using the 3D object information when such information is available. This is different from simply calculating the underlying disparity and can be quite helpful for the disambiguation of the target selection when the objects are not close enough to the viewer (or far enough from each other) to be distinguished by the amount of the disparity. In this situation, the closest 3D object (or 3D object element) to the estimated 3D cursor position may be prioritized as the targeted object.

10.2 Practical Application of Stereo Cursor

The stereo 3D mouse cursor can be used in a wide range of applications, including 3D computer games and 3D object manipulation. To examine the applicability of the stereo cursor in manipulating 3D objects, we developed a simple 3D object manipulation toolset for MVSplat. The toolset contains a pen and an eraser tool with a few auxiliary displaying-state control buttons which together allow the user to highlight a desired 3D point of the object or remove the tag of the previously highlighted points within the virtual stereoscopic 3D space.

Figure 10.4 shows a sample result of applying the 3D editing toolset (automatic mode) for ground-truth data measurement. Here the S3DCursor is used to specify a 3D-contour surrounding a TB cavity on a 3D lung model. The red-blue mode of

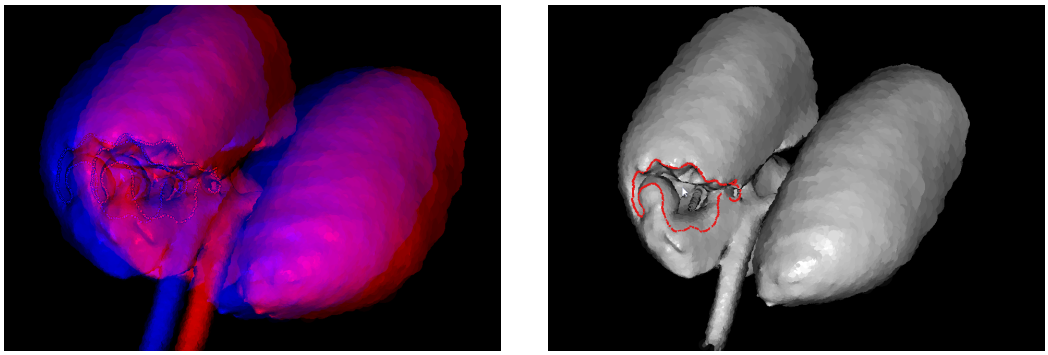


Figure 10.4: Marking the TB cavity boundaries on a 3D lung model using stereo 3D toolset. Left: red-blue stereoscopic representation mode. Right: ordinary perspective representation.

Table 10.1: S3DCursor user evaluation criteria.

Experiment	**Criterion**
Experiment 1	**C1:** Stereo mode gives better 3D visual experience
	C2: Stereo cursor helps in having better 3D experience
Experiment 2	**C3:** Object (boundaries) are better distinguishable in stereo mode
	C4: Stereo cursor improves distinguishability
Overall	**C5:** Simplicity of using stereo cursor
	C6: Satisfaction and usefulness of the stereo cursor

MVSplat is used in order to provide the stereo effect of the model in a virtual 3D environment. The left image, Figure 10.4a, shows the red-blue representation of the model and the resulting 3D contour. The right image, Figure 10.4b, shows the corresponding single-view version of the model and the contour that is generated in the red-blue visualization mode. Although these images are degraded due to down-scaling, it should be still possible to watch the stereo effect in 3D using simple anaglyph glasses and compare it with the corresponding 2D image.

10.2.1 User Evaluation of Stereo Cursor

We used MVSplat and its editing toolset to evaluate the performance of 3D object manipulation tasks within a stereoscopic environment versus doing the same tasks on the perspective projection of the object on a conventional 2D display. Again the red-blue mode of MVSplat was used to provide the stereo effect. We conducted two sets of experiments. Table 10.1 shows a brief description of the criteria used in these

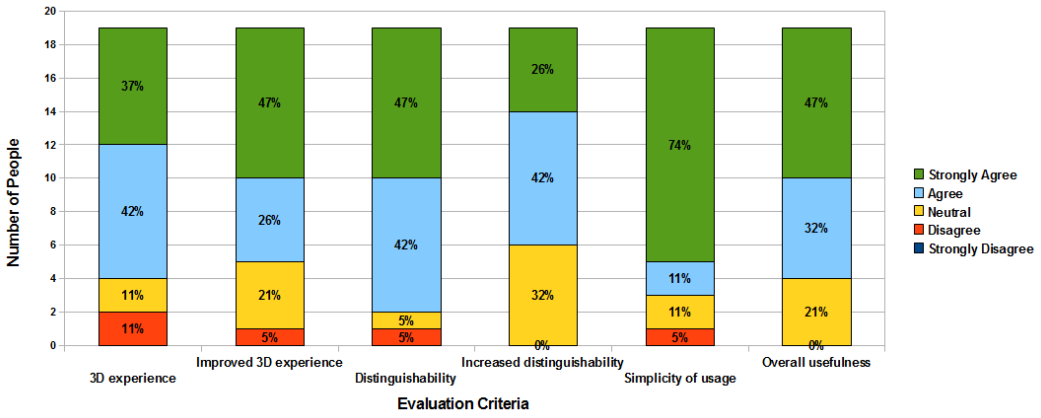


Figure 10.5: The stereo cursor user evaluation results.

experiments. None of people who participated in these experiments had previous experience working with the stereo cursor, but many of them had the experience of watching stereo content in 3D at least once. Moreover, some of them were already familiar with some 3D input devices, such as the space navigator, and a few of them were familiar with some line cursors that are used to point to a 3D location within the CAVE VR systems [105].

In the first experiment the subjects were asked to compare their visual experience with 3D stereo visualization versus 2D visualization. The purpose of this experiment was to show whether the ability of virtually touching the surface of the objects using S3DCursor gives a better understanding of their 3D shape to the people. Three different models *i.e.* Lucy, Dragon, and Donna were randomly chosen and used in performing this experiment. As the first criterion in this experiment, we wanted to make sure that the participants are able to observe the 3D effects of the stereo using anaglyph glasses and also to separate the 3D feeling caused by stereo from those possible 3D experiences that may be caused by 3D movements of the cursor. Even though the red-blue representation usually causes some ghosting effects, the majority of people (strongly) agreed that they have better 3D experience with the stereo mode than the ordinary single view rendering mode before using the 3D cursor (see Figure 10.5, column 1). The second criterion asks the subjects if the S3DCursor helps in having a better 3D visual experience when they use it to touch different parts of the object's surface. Again the majority of users believe that when they slide the cursor over the 3D object, the depth movements of the 3D cursor give them a better sense of the 3D shape of the object (Figure 10.5, column 2). According to these results, unlike a 2D cursor which may cause some disturbance within

the stereoscopic 3D environment, the S3DCursor helps to provide a better visual experience by giving the users the interesting feeling of touching the surface of the virtual 3D object.

In the second experiment we asked the user to perform the same manipulation tasks on the 3D Lung model in both single and stereo view modes. In fact, we asked them to mark the boundaries of a TB cavity in both 2D and 3D representations. Again our user survey suggests that the stereo cursor provides better depth information (Figure 10.5, column 3) and that users can more accurately specify the region of interest than when using the conventional 2D cursor on the 2D display (Figure 10.5, column 4). In fact, stereo 3D visualization plus S3DCursor movements allow the user to better discriminate the concavities and convexities of the object surface, and hence the border line of the TB cavity.

Finally, we asked the users to express their overall assessment of the technique in terms of the simplicity of usage and also its usability based on their short experience of working with virtual 3D objects using stereo cursor. The majority of the participants strongly agreed that S3DCursor can be used as easily as an ordinary 2D mouse cursor and many of them found it a useful method for selecting and manipulating 3D objects.

Chapter 11

Conclusion

In this thesis we reported on our exploration in the area of stereo-based 3D reconstruction and visualization. We looked into a variety of different problems involved in the process of capturing, transmitting, and displaying (stereo) 3D content. The problems we studied in this thesis, while they were different in terms of approach and focus, all had one aspect in common, *i.e.*, dealing with the discretized (sampled) representations composed of, for example, picture elements (pixels), volume elements (voxels), surface elements (surfels), or in general discrete 2D/3D points or samples. In this regard, we studied methods of optimal discretization of stereo images for 3D reconstruction and 3D visualization. We also presented algorithms, techniques, and data structures for compact representation and interactive visualization of point-based 3D objects for conventional and stereo-based 3D applications, as well as methods of interaction and manipulation of such visualized objects. We organized the results of our work and studies in three parts highlighting the main directions and focus of our research.

11.1 Part I - Summary

In Part I we described a theoretical model for determining the trade-off between vertical and horizontal resolutions for stereo-based 3D reconstruction and 3D viewing applications. We inferred optimal pixel aspect ratios (PARs) by considering two commonly used stereo setups, parallel and with-vergence configurations, and showed how optimal PAR is related to the corresponding stereo configuration parameters. We extended the results to the case of stereo viewing on stereoscopic 3D displays by deriving a model of stereo viewing that takes some parameters such as display viewing distance and human vision specifications into account, and used

this model to optimize the pixel aspect ratio for 3D viewing on a stereoscopic 3D display. Using the formulations derived we showed that, in general, for a given total resolution a more dense distribution of pixels along the horizontal direction often improves the accuracy of the 3D estimation for both configurations, although the optimal PAR may vary considerably with different parameter values. Applying more practical parameter values to the inferred equations, our theoretic suggests that a pixel aspect ratio of 2:3 (pixel-width:pixel-height) will lead to a better stereo-based 3D reconstruction.

We further extended the results by proposing a unified model of optimal discretization for both stereo-based 3D reconstruction and stereo 3D-viewing applications. We derived this general solution for optimal PAR as a function of the display or image plane aspect ratio and the range of plausible disparity values. This general theoretical model reaffirms that, having a fixed total resolution, a more horizontally dense discretization of the imaging sensor or display screen of the stereo device leads to a more accurate 3D reconstruction as well as a better stereo-based 3D viewing experience irrespective of the stereo capturing or viewing configuration parameters. Based on this general model, we again suggest a pixel aspect ratio of about 2:3 for both stereo capturing and viewing ends. This extended formulation does not enforce any constraint on capturing or viewing stereo content, such as enforcing a specific capturing or viewing distance, or using fixed, predefined values for camera focal length and stereo base-line. This means that this solution can conveniently be applied in the design and manufacture of digital stereo-based 3D capture and display devices to achieve more accurate 3D reconstruction and enhanced 3D viewing experience.

We validated the above-mentioned theoretical results through subjective user evaluations on both synthetic and real stereo content of different simulated pixel aspect ratios in which groups of physical pixels were used to form a pixel of the desired aspect ratio. Our experiments on these simulated images showed that from a subjective point of view, the suggested optimal PAR, 2:3, indeed improves the 3D output.

11.2 Part I - Future Extensions

There are several directions in which this part of our research can be extended in the future:

First of all, our method of simulating different PARs using pixel grouping tech-

nique introduces noticeable blockiness into images. In this regard, it would be interesting to incorporate the concept of Just Noticeable Difference (JND) for 3D perception [26] to see how the results may change. We expect that, as with HDTV displays, the whole effect of applying optimal PAR on a high-resolution 3D display results in noticeable improvement on the overall 3D output quality, even though the user may not be able to separately distinguish the individual pixels of such high-resolution displays.

It will also be interesting to extend these studies to the case of downsizing (subsampling) stereo images/videos apart from the shape of the physical pixel of the media, which is generally assumed to be square on current digital devices. Considering our theoretical findings and user studies, we expect that keeping more information on the horizontal axis would better preserve the 3D content of the scene. Upon conformance, the idea can be very useful in the compression and transmission of the stereo content, especially on networks with limited bandwidth.

The mathematical optimization models described in this part all have been based on minimizing the maximum (relative) estimation error in the vertical position of a 3D point. This method of optimization implicitly involves horizontal and depth components and has the advantage of preventing arbitrary degradation of the estimation in the vertical direction. However it would be interesting to study the problem with respect to some other cost functions such as Euclidean distance of the original and estimated 3D point, or the volume of the 3D voxel formed by two corresponding pixels in the stereoscopic 3D space. In this regard, different human perceptual factors [12, 25, 28, 79], motion factors [29], and error measures produced by cross-talk, correlation, and epipolar constraints [19, 6] may also be taken into account in the hypothesized cost function. Such extensions may also take into account the relative importance of vertical *vs.* horizontal disparities in the human visual system [84, 31], and may incorporate an image quality metric to reflect the picture quality changes caused by dissimilar horizontal and vertical discretization.

Finally, our model can be studied with regard to the spatially varying distribution of the cones in the human eyes [87] or in combination with the subsampling models and anti-alias filters presented in [64] for further enhancement of the output of multi-view autostereoscopic 3D displays.

11.3 Part II - Summary

In Part II we introduced an implicit tree structure, called Multi-Section tree, for balanced representation of an arbitrary number of points with the capability of distinguishing between internal nodes and leaves, even if they may not be of the same size or may be shuffled at the penultimate level. We showed how this structure, together with our enhanced 3D/4D hierarchical quantization schemes, can efficiently be used in representation and rendering of both still and animated point-based 3D models. Specifically, we used this structure to build a two-layer, fully-balanced hierarchy over the 3D samples representing a still 3D object. This two-layer representation is very compact and is more accurate than the state-of-the-art algorithms, offering the possibility of adjusting the level of accuracy, and as a result the level of rendering quality, depending on the targeted application. It also works very well in handling shapes with complex, highly-detailed surface structures. We demonstrated the efficiency of our balanced structure for the case of moving 3D objects, as well. In this regard, we introduced algorithms and techniques for appropriate distribution of the spatio-temporal 4D samples representing a dynamic 3D object within the introduced balanced tree structure. We also introduced an interactive environment for animated rendering and simultaneous spatial and temporal navigation of the moving 3D object. Finally, we discussed a streaming and rendering architecture based on which our current implementation is applicable to long sequences of 3D frames with a large number of samples in each frame.

11.4 Part II - Future Extensions

The two-layer structure, which we have implemented to represent still 3D objects, imposes some sort of regularity on the point-based 3D model by dividing its surface into a number of equally sized, disjointed patches. We are considering extending this to a multi-layer structure to improve the speed and quality of the rendering, and the compactness of the representation. We are also thinking of extending this layered structure to the case of animated objects. This regularity can also be employed in enhancing the compression ratio of color 3D models, as it potentially allows the frequency domain encoding schemes, such as the DCT or Wavelet transform, to be used in the compression of the point colors and other similar attributes. In addition, the possibility of using on-the-fly Delaunay triangulation on the small patches of neighboring samples may eliminate the necessity of storing some or all of the point

normals. This can result in significant reduction in the file size at the expense of some, probably negligible, processing overhead during the rendering. These regular patches also introduce some sort of implicit connectivity, in the lack of explicit connectivity information, which can be very useful in implementing empirical or physical deformations and manipulations on the 3D object.

As another extension of our work in this area, a hybrid of our representation and other compact structures such as [18] and [55] may be considered to more effectively use the points' similarities in subpartitions to obtain even more compact representations.

Finally, we are thinking of using of our spatio-temporal interactive rendering implementation in a virtual stereoscopic 3D space. This interactive environment, combined with the object selection and manipulation tools we have implemented in Part III, can be used in many interesting applications such as virtual motion capture, object motion correction, or even semi-automatic frame interpolation.

11.5 Part III - Summary

In Part III we discussed different theoretical aspects of a stereo-based 3D pointing technique which enables the users to point to an arbitrary location inside the 3D space projected by a stereoscopic 3D display, and we described our approach to implementing such a 3D cursor. We also discussed how such a technique satisfies all the requirements of an abstract 3D cursor so that it potentially can be considered as a simple extension of the 2D cursor to the stereoscopic 3D space. The user experience with the stereo 3D cursor shows that it has great potential for carrying the most common interaction and manipulation tasks within the virtual space. In this thesis we showed its usefulness and efficiency in working with 3D objects and presented some promising results.

11.6 Part III - Future Extensions

Further improvement is necessary to overcome some drawbacks of the stereo cursor like its low accuracy in the farther depths. This can probably be achieved by implementing some of the suggested virtual enhancements. The optimal pixel aspect ratio model, described in Part I, can also greatly contribute to the improvement of the resolution along the depth dimension. However, this improvement is conditional upon the physical existence of such stereo-optimized 3D display devices. We also

need to improve our UI tools for smarter manipulation of 3D contours, surfaces, and volumes based on some shape or shade analysis algorithms, and also using the capabilities of the more advanced stereoscopic 3D displays and 3D input devices.

Bibliography

[1] Biomedcache user guide. Software User Guide, Available at: `http://www.ch.ic.ac.uk/local/organic/medchem/UserGuide.pdf`, Februray 2nd 2005. Date of Access: August 24, 2009.

[2] OrthoEngine 3D stereo. Technical Specification, Available at: `http://www.pcigeomatics.com/pdfs/3D_Stereo.pdf`, Februray 2nd 2005. Date of Access: August 24, 2009.

[3] B. Adams. *Point-Based Modeling, Animation and Rendering of Dynamic Objects*. PhD thesis, Informatics Section, Department of Computer Science, Faculty of Engineering, May 2006. Dutré, Philip (supervisor), Willems, Yves (cosupervisor).

[4] E. H. Adelson and J. R. Bergen. The Plenoptic Function and the Elements of Early Vision. *Computational Models of Visual Processing*, pages 3–20, 1991.

[5] S. J. Adelson and C. D. Hansen. Fast stereoscopic images with ray-traced volume rendering. In *VVS '94: Proceedings of the 1994 symposium on Volume visualization*, pages 3–9, New York, NY, USA, 1994. ACM.

[6] N. J. Ayache. *Artificial Vision for Mobile Robots: Stereo Vision and Multisensory Perception*. MIT Press, 1991.

[7] H. Azari, I. Cheng, and A. Basu. Optimal pixel aspect ratio for stereoscopic 3D displays under practical viewing conditions. In *3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video, 2009*, pages 1 –4, May 2009.

[8] H. Azari, I. Cheng, and A. Basu. Stereo 3D mouse (S3D-Mouse): Measuring ground truth for medical data in a virtual 3D space. *Conf Proc IEEE Eng Med Biol Soc*, 1:5744–7, 2009.

[9] H. Azari, I. Cheng, and A. Basu. Stereo 3D mouse cursor: A method for interaction with 3D objects in a stereoscopic virtual 3D space. *Int. J. Digital Multimedia Broadcasting*, 2010, 2010.

[10] H. Azari, I. Cheng, and A. Basu. Optimized point splatting based on a fully-balanced hierarchical structure. In *Proceedings of IEEE International Conference on Multimedia and Expo*, 2011.

[11] H. Azari, I. Cheng, K. Daniilidis, and A. Basu. Optimal pixel aspect ratio for enhanced 3D TV visualization. *Comput. Vis. Image Underst.*, 116(1):38–53, Jan 2012.

[12] B. T. Backus, D. J. Fleet, A. J. Parker, and D. J. Heeger. Human Cortical Activity Correlates With Stereoscopic Depth Perception. *J Neurophysiol*, 86(4):2054–2068, Oct 2001.

[13] S. Baker, R. Szeliski, and P. Anandan. A layered approach to stereo reconstruction. In *Computer Vision and Pattern Recognition, 1998. Proceedings. 1998 IEEE Computer Society Conference on*, pages 434–441, Jun 1998.

[14] A. Basu. Optimal discretization for stereo reconstruction. *Pattern Recognition Letters*, 13(11):813–820, 1992.

[15] A. Basu and H. Sahabi. Optimal non-uniform discretization for stereo reconstruction. In *Pattern Recognition, 1996., Proceedings of the 13th International Conference on*, volume 1, pages 755–759 vol.1, Aug 1996.

[16] C. V. Berkel and J. A. Clarke. Characterization and optimization of 3D-LCD module design. In Scott S. Fisher, John O. Merritt, and Mark T. Bolas, editors, *Stereoscopic Displays and Virtual Reality Systems IV*, volume 3012, pages 179–186. SPIE, 1997.

[17] F. Bettio, E. Gobbetti, F. Marton, A. Tinti, E. Merella, and R. Combet. A point-based system for local and remote exploration of dense 3D scanned models. In *The 10th Int. Symp. on Virtual Reality, Archaeology and Cultural Heritage*, Oct 2009.

[18] M. Botsch, A. Wiratanaya, and L. Kobbelt. Efficient high quality rendering of point sampled geometry. In *EGRW '02: Proceedings of the 13th Eurographics workshop on Rendering*, pages 53–64, Aire-la-Ville, Switzerland, Switzerland, 2002. Eurographics Association.

[19] T. Brodský, C. Fermüller, and Y. Aloimonos. Structure from motion: Beyond the epipolar constraint. *International Journal of Computer Vision*, 37(3):231–258, 2000.

[20] M. Z. Brown, D. Burschka, and G. D. Hager. Advances in computational stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25:993–1008, 2003.

[21] E. E. Catmull. *A Subdivision Algorithm for Computer Display of Curved Surfaces*. PhD thesis, Department of Computer Science, University of Utah, December 1974.

[22] B. Chen and M. X. Nguyen. Pop: A hybrid point and polygon rendering system for large data. *Visualization Conference, IEEE*, 0:null, 2001.

[23] M. Chen, C. Correa, S. Islam, M. W. Jones, P.-Y. Shen, D. Silver, S. J. Walton, and P. J. Willis. Manipulating, deforming and animating sampled object representations. *Computer Graphics Forum*, 26(4):824–852, 2007.

[24] I. Cheng and A. Basu. Optimal aspect ratio for 3D TV. In *3DTV Conference, 2007*, pages 1–4, May 2007.

[25] I. Cheng and A. Basu. Perceptually optimized 3-D transmission over wireless networks. *Multimedia, IEEE Transactions on*, 9(2):386–396, Feb. 2007.

[26] I. Cheng and P. Boulanger. A 3D perceptual metric using just-noticeable-difference. In *In Eurographics Short Presentations*, pages 97–100, 2005.

[27] I. Cheng, K. Daniilidis, and A. Basu. Optimal aspect ratio under vergence for 3D TV. In *3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video, 2008*, pages 209–212, May 2008.

[28] L. F. Cheong. Depth perception under motion and stereo with implications for 3D TV. In *3DTV Conference, 2007*, pages 1–4, May 2007.

[29] L. F. Cheong, C. Fermüller, and Y. Aloimonos. Effects of errors in the viewing geometry on shape estimation. *Computer Vision and Image Understanding*, 71(3):356–372, 1998.

[30] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. 3rd edition edition, 2009.

[31] V. Cornilleau-Pérès and J. Droulez. Visual perception of surface curvature: psychophysics of curvature detection induced by motion parallax. *Perception & psychophysics*, 46(4):351–364, Oct 1989.

[32] B. G. Cumming and A. J. Parker. Binocular neurons in V1 of awake monkeys are selective for absolute, not relative, disparity. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 19(13):5602–5618, Jul 1999.

[33] N. T. Dang. A survey and classification of 3D pointing techniques. In *Research, Innovation and Vision for the Future, 2007 IEEE International Conference on*, pages 71 –80, March 2007.

[34] T. K. Dey, J. Giesen, and J. Hudson. Decimating samples for mesh simplification. In *Proc. 13th Canadian Conf. Comput. Geom*, pages 85–88, 2001.

[35] T. K. Dey, J. Giesen, and J. Hudson. Delaunay based shape reconstruction from large data. In *PVG '01: Proceedings of the IEEE 2001 symposium on parallel and large-data visualization and graphics*, pages 19–27, Piscataway, NJ, USA, 2001. IEEE Press.

[36] T. K. Dey and J. Hudson. Pmr: point to mesh rendering, a feature-based approach. In *VIS '02: Proceedings of the conference on Visualization '02*, pages 155–162, Washington, DC, USA, 2002. IEEE Computer Society.

[37] N. A. Dodgson. Autostereoscopic 3D displays. *Computer*, 38(8):31–66, Aug 2005.

[38] D. S. Ebert, C. D. Shaw, A. Zwa, and C. Starr. Two-handed interactive stereoscopic visualization. In *PROCEEDINGS IEEE VISUALIZATION '96*, pages 205–210, 1996.

[39] B. J. Frey and N. Jojic. A comparison of algorithms for inference and learning in probabilistic graphical models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(9):1392–1416, 2005.

[40] B. Frohlich, J. Hochstrate, A. Kulik, and A. Huckauf. On 3D input devices. *Computer Graphics and Applications, IEEE*, 26(2):15 – 19, March–April 2006.

[41] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

[42] D. Gerszewski, H. Bhattacharya, and A. W. Bargteil. A point-based method for animating elastoplastic solids. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, Aug 2009.

[43] E. Gobbetti and F. Marton. Layered point clouds: a simple and efficient multiresolution structure for distributing and rendering gigantic point-sampled models. *Computers and Graphics*, 28(6):815 – 826, 2004.

[44] P. Goswami, Y. Zhang, R. Pajarola, and E. Gobbetti. High quality interactive rendering of massive point models using multi-way kd-trees. In *Proc. Pacific Graphics*, 2010.

[45] C. Gotsman. On graph partitioning, spectral analysis, and digital mesh processing. In *In Proc. Intl. Conf. Shape Modeling and Applications*, pages 165–174. IEEE Computer Society, 2003.

[46] J. P. Grossman and W. J. Dally. Point sample rendering. In *19th Eurographics Workshop on Rendering Techniques*, pages 181–192. Springer, 1998.

[47] C. Hand. A survey of 3-D input devices. Technical Report TR94/2, Department of Computer Science, De Montfort University, UK, 1994.

[48] C. Hand. A survey of 3D interaction techniques. In *Computer Graphics Forum*, volume 16, pages 269–281, 1997.

[49] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, March 2004.

[50] T. He and A. E. Kaufman. Fast stereo volume rendering. In *IEEE Visualization*, pages 49–56, 1996.

[51] K. E. Hoff. Deriving the opengl perspective depth transformation. Technical Report, Available at: `http://www.cs.unc.edu/~geom/HOFF/techrep/perspective.doc`, Spring 2000. Date of Access: August 18, 2009.

[52] N. Holliman. 3D display systems. Department of Computer Science, University of Durham, `http://www.dur.ac.uk/n.s.holliman/Presentations/3dv3-0.pdf`, Februray 2nd 2005. Date of Access: August 18, 2009.

[53] T. Hübner and R. Pajarola. Single-pass multi-view volume rendering. In *Proceedings IADIS International Conference Computer Graphics and Visualization*, 2007.

[54] T. Hübner, Y. Zhang, and R. Pajarola. Multi-view point splatting. In *GRAPHITE '06: Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia*, pages 285–294, New York, NY, USA, 2006. ACM.

[55] E. Hubo, T. Mertens, T. Haber, and P. Bekaert. The quantized kd-tree: Efficient ray tracing of compressed point clouds. In *Interactive Ray Tracing 2006, IEEE Symposium on*, pages 105 –113, Sept. 2006.

[56] B. Javidi and F. Okano. *Three-dimensional television, video, and display technologies*. Physics and Astronomy Online Library. Springer, 2002.

[57] H. W. Jensen. *Realistic image synthesis using photon mapping*. A. K. Peters, Ltd., Natick, MA, USA, 2001.

[58] A. Kalaiah and A. Varshney. Statistical geometry representation for efficient transmission and rendering. *ACM Trans. Graph.*, 24(2):348–373, 2005.

[59] S. B. Kang, R. Szeliski, and J. Chai. Handling occlusions in dense multi-view stereo. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–103 – I–110 vol.1, 2001.

[60] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *The Bell System Technical Journal*, 49(1):291–307, 1970.

[61] S. Kiyohide and O. Yuichi. Passive depth acquisition for 3D image displays (special issue on 3D image processing). *IEICE transactions on information and systems*, 77(9):949–957, 1994-09-25.

[62] L. Kobbelt and M. Botsch. A survey of point-based techniques in computer graphics. *Computers and Graphics*, 28(6):801 – 814, 2004.

[63] V. Kolmogorov and R. Zabih. Multi-camera scene reconstruction via graph cuts. pages 82–96, 2002.

[64] J. Konrad and P. Agniel. Subsampling models and anti-alias filters for 3-D automultiscopic displays. *Image Processing, IEEE Transactions on*, 15(1):128 –140, Jan. 2006.

[65] J. Konrad and M. Halle. 3-D displays and signal processing. *Signal Processing Magazine, IEEE*, 24(6):97 –111, Nov. 2007.

[66] Y. M. Koo, C. H. Lee, and Y. G. Shin. Object-order template-based approach for stereoscopic volume rendering. *Journal of Visualization and Computer Animation*, 10(3):133–142, 1999.

[67] S. Kumar, D. Manocha, W. F. Garrett, and M. C. Lin. Hierarchical back-face computation. In *Rendering Techniques'96*, pages 235–244, 1996.

[68] T. Kumar and D. A. Glaser. Depth discrimination of a line is improved by adding other nearby lines. *Vision Research*, 32(9):1667 – 1676, 1992.

[69] M. Levoy and T. Whitted. The use of points as a display primitive. Technical Report 85-022, Computer Science Department, University of North Carolina at Chapel Hill, January 1985.

[70] S. Mantler and A. L. Fuhrmann. Point based rendering for massive data sets: A case study. *Computer Graphics International Conference*, 0:182–187, 2004.

[71] L. Matthies and S. Shafer. Error modeling in stereo navigation. *Robotics and Automation, IEEE Journal of*, 3(3):239 –248, June 1987.

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

[73] D. F. Mcallister. *Stereo computer graphics and other true 3D technologies*. Princeton University Press, 1993.

[74] D. F. McAllister. *Stereo Computer Graphics and other True 3D Technologies*. Princeton U. Press, Princeton, NJ, 1993.

[75] D. F. McAllister. *3D Displays*, pages 1327–1344. Wiley Encyclopedia on Imaging, 2002.

[76] T. Mitsuhashi. A study of the relationship between scanning specifications and picture quality. NHK, NHK Laboratories Note, no. 256, October 1980.

[77] K. Museth and S. Lombeyda. Tetsplat real-time rendering and volume clipping of large unstructured tetrahedral meshes. *Visualization Conference, IEEE*, 0:433–440, 2004.

[78] W. C. Navidi. *Statistics for engineers and scientists, 2nd edition*. Boston: McGraw-Hill Higher Education, 2008.

[79] Y. Pan, I. Cheng, and A. Basu. Quality metric for approximating subjective evaluation of 3-D objects. *Multimedia, IEEE Transactions on*, 7(2):269–279, April 2005.

[80] S. B. Park and S. U. Lee. Multiscale representation and compression of 3-D point data. *IEEE Transactions on Multimedia*, 11(1):177–183, 2009.

[81] M. Pauly, R. Keiser, B. Adams, P. Dutré, M. Gross, and L. J. Guibas. Meshless animation of fracturing solids. In *ACM SIGGRAPH 2005 Papers*, SIGGRAPH '05, pages 957–964, New York, NY, USA, 2005. ACM.

[82] H. Pfister, M. Zwicker, J. van Baar, and M. Gross. Surfels: surface elements as rendering primitives. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 335–342, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[83] Y. Quan, W. H. Li, Y. J. Pang, and G. J. Liu. An efficient point rendering system for ray tracing. In *Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on*, volume 9, pages 5429 –5431 Vol. 9, Aug. 2005.

[84] B. J. Rogers and M. E. Graham. Anisotropies in the perception of three-dimensional surfaces. *Science*, 221(4618):1409–1411, 1983.

[85] S. Rusinkiewicz and M. Levoy. Qsplat: a multiresolution point rendering system for large meshes. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 343–352, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[86] S. A. Safier and M. W. Siegel. 3D-Stereoscopic X Windows. In *Proceedings of the 1995 SPIE/IS+T Conference (San Jose*, pages 160–167, 1995.

[87] H. Sahabi and A. Basu. Analysis of error in depth perception with vergence and spatially varying sensing. *Computer Vision and Image Understanding*, 63(3):447–461, 1996.

[88] M. Sainz and R. Pajarola. Point-based rendering techniques. *Comput. Graph.*, 28(6):869–879, 2004.

[89] N. Salman and M. Yvinec. Surface Reconstruction from Multi-View Stereo of Large-Scale Outdoor Scenes. *International Journal of Virtual Reality*, Volume 9(Number 1):19–26, Mar 2010. Special issue for ACCV09.

[90] H. Samet. Applications of spatial data structures. Addison-Wesley, 1990.

[91] D. Scharstein, R. Szeliski, and R. Zabih. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. In *Stereo and Multi-Baseline Vision, 2001. (SMBV 2001). Proceedings. IEEE Workshop on*, pages 131–140, 2001.

[92] O. Schreer, P. Kauff, and T. Sikora, editors. *3D Videocommunication: Algorithms, concepts and real-time systems in human centred communication*. Wiley, 2005.

[93] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 519 – 528, June 2006.

[94] P. J. H. Seuntins. *Visual experience of 3D TV*. PhD thesis, Eindhoven: Technische Universiteit Eindhoven, 2006.

[95] J. Y. Sim, C. S. Kim, and S. U. Lee. Lossless compression of 3-D point data in qsplat representation. *IEEE Transactions on Multimedia*, 7(6):1191–1195, 2005.

[96] J. Y. Sim and S. U. Lee. Compression of 3-D point visual data using vector quantization and rate-distortion optimization. *IEEE Transactions on Multimedia*, 10(3):305–315, 2008.

[97] F. Steinicke, G. Bruder, K. H. Hinrichs, and T. Ropinski. Simultane 2D/3D user interface konzepte für autostereoskopische desktop-vr systeme. In *4. GI-Workshop AR/VR*, pages 125–132. GI-Fachgruppe VR/AR, Shaker-Verlag, 2007.

[98] J. Sun, N. N. Zheng, and H. Y. Shum. Stereo matching using belief propagation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(7):787 – 800, July 2003.

[99] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother. A comparative study of energy minimization methods for markov random fields with smoothness-based priors. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(6):1068–1080, 2008.

[100] D. Vlasic, I. Baran, W. Matusik, and J. Popović. Articulated mesh animation from multi-view silhouettes. *ACM Trans. Graph.*, 27:97:1–97:9, August 2008.

[101] M. Wan, N. Zhang, H. Qu, and A. E. Kaufman. Interactive stereoscopic rendering of volumetric environments. *IEEE Transactions on Visualization and Computer Graphics*, 10:15–28, 2004.

[102] G. Westheimer. Cooperative neural processes involved in stereoscopic acuity. *Experimental Brain Research*, 36:585–597, 1979. 10.1007/BF00238525.

[103] J. S. Yedidia, W. T. Freeman, and Y. Weiss. Understanding belief propagation and its generalizations. *Exploring artificial intelligence in the new millennium*, pages 239–269, 2003.

[104] Z. Yordanov. *Optimal sub-pixel arrangements and coding for ultra-high resolution three-dimensional OLED displays*. Kassel, 2007.

[105] S. Zhang, C. Demiralp, D. F. Keefe, M. DaSilva, D. H. Laidlaw, B. D. Greenberg, P. J. Basser, C. Pierpaoli, E. A. Chiocca, and T. S. Deisboeck. An immersive virtual environment for dt-mri volume visualization applications: a case study. In *Visualization, 2001. VIS '01. Proceedings*, pages 437 –584, Oct. 2001.

[106] Y. Zhou and M. Garland. Interactive point-based rendering of higher-order tetrahedral data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1229–1236, 2006.

[107] M. Zwicker, M. H. Gross, and H. Pfister. A survey and classification of real time rendering methods. Technical Report No. 332, Computer Science Department, ETH Zürich, 1999.