EFFICIENT RENDERING FOR THREE-DIMENSIONAL DISPLAYS

Djamel Hassaine

A Thesis presented for the degree of
Doctor of Philosophy

School of Engineering and Computing Sciences

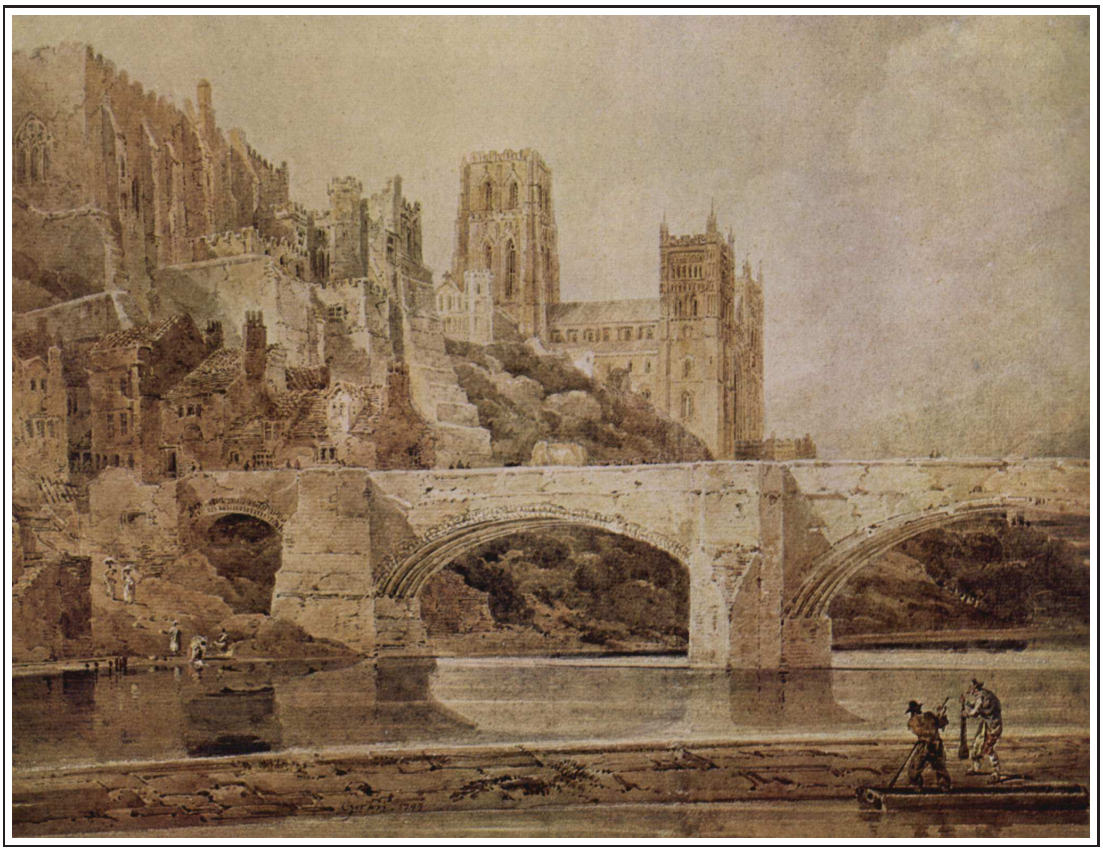
Durham University

United Kingdom

2010

For my parents,

Yamani and Lesley



Thomas Girtin - Durham Cathedral 1799, from [50]

Abstract

This thesis explores more efficient methods for visualizing point data sets on three-dimensional (3D) displays. Point data sets are used in many scientific applications, e.g. cosmological simulations. Visualizing these data sets in 3D is desirable because it can more readily reveal structure and unknown phenomena. However, cutting-edge scientific point data sets are very large and producing/rendering even a single image is expensive. Furthermore, current literature suggests that the ideal number of views for 3D (multiview) displays can be in the hundreds, which compounds the costs.

The accepted notion that many views are required for 3D displays is challenged by carrying out a novel human factor trials study. The results suggest that humans are actually surprisingly insensitive to the number of viewpoints with regard to their task performance, when occlusion in the scene is not a dominant factor.

Existing stereoscopic rendering algorithms can have high set-up costs which limits their use and none are tuned for uncorrelated 3D point rendering. This thesis shows that it is possible to improve rendering speeds for a low number of views by perspective reprojection. The novelty in the approach described lies in delaying the reprojection and generation of the viewpoints until the fragment stage of the pipeline and streamlining the rendering pipeline for points only. Theoretical analysis suggests a fragment reprojection scheme will render at least 2.8 times faster than naïvely re-rendering the scene from multiple viewpoints.

Building upon the fragment reprojection technique, further rendering performance is shown to be possible (at the cost of some rendering accuracy) by restricting the amount of reprojection required according to the stereoscopic resolution of the display. A significant benefit is that the scene depth can be mapped arbitrarily to the perceived depth range of the display at no extra cost than a single region mapping approach. Using an average case-study (rendering from a 500k points for a 9-view High Definition 3D display), theoretical analysis suggests that this new approach is capable of twice the performance gains than simply reprojecting every single fragment, and quantitative measures show the algorithm to be 5 times faster than a naïve rendering approach. Further detailed quantitative results, under varying scenarios, are provided and discussed.

Declaration

The work in this thesis is based on research carried out at the Innovative Computing Group, School of Engineering and Computing Sciences, Durham University. No part of this thesis has been submitted elsewhere for any other degree or qualification and all of it is my own work unless referenced to the contrary in the text.

Publications

Sections of the work contained in Chapter 3 have been published in ACM Transactions on Applied Perception (TAP), Volume 8, Issue 1, October 2010, “Investigating the performance of path searching tasks in depth on multiview displays”.

Acknowledgements

I would like to express my heartfelt gratitude to my parents, Yamani and Lesley, who encouraged me to study for this PhD, which has greatly influenced my life and the way I think.

I give thanks and love to: my partner Emma; my uncle and aunt, Alan and Sue; and my grandmother Ruth.

I am grateful for the advice and support from my supervisor, Dr. Nicolas Holliman; Prof Simon Liversedge for his invaluable advice on statistics and psychology; and Barbara Froner, Hazel Blythe, Paul Gorley and Geng Sun for their contributions to this thesis.

Financial support was generously provided by Durham University's Department of Computer Science and Department of Psychology, and of course my parents.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter the basic background and motivation for addressing the issues of visualizing large point data sets using multiview displays are provided.

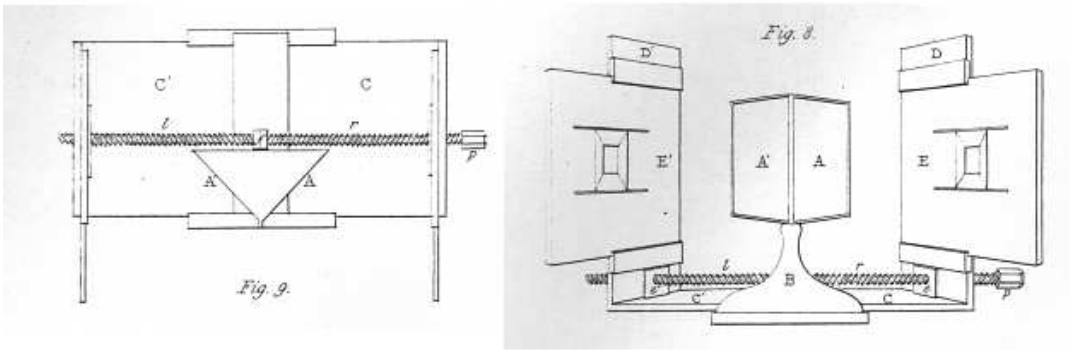
1.1 Stereoscopy

In 1838, Charles Wheastone [166] demonstrated, with his stereoscope, that two images captured from a different horizontal centre of projection will invoke a powerful and unique depth sense when presented to each eye; this phenomenon is known as stereopsis (literally "solid seeing"). Since then, numerous inventions have been developed to capture, generate and display three-dimensional (3D) stereoscopic scenes.

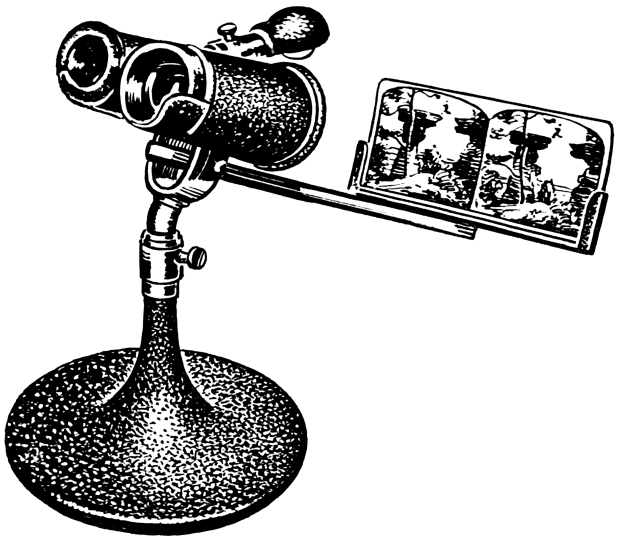
Wheatstone's stereoscope (illustrated in Figure 1.1) consisted of two mirrors angled 90° to each other which reflected a different image to each eye. His images, which were hand drawn, were mounted on sliding boards controlled by a wooden screw that allowed the observer to adjust the distance of the images until the two reflected images coincided at the intersection of the optic axes. Although the stereoscope was crude in design, it was the first ever known device to provide scientific proof of a link between binocular vision and depth perception.

Binocular depth perception is quite subtle and if the reader is unfamiliar with the concept, he or she may be skeptical of any improvement in depth perception emanating from binocular vision (try closing one eye and see if you can notice any difference). When Wheatstone first proposed binocular disparity as a depth cue, many well-respected

Figure 1.1: Original diagrams of Wheatstone's stereoscope (from [166]) and a drawing of a later and improved Holmes type stereoscope, from [48].



(a) Wheatsone's stereoscope

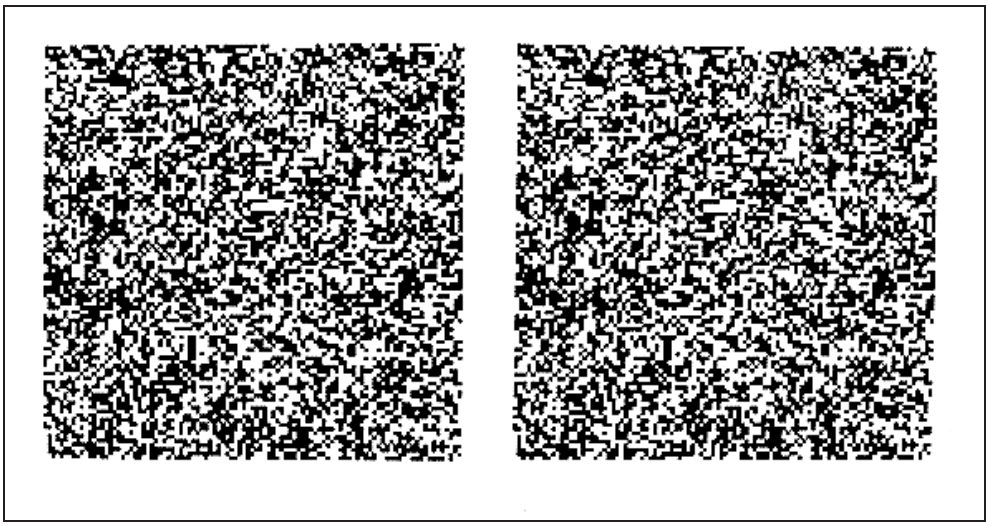


(b) Improved Holmes stereoscope design

scientists were also skeptical, in fact Sir David Brewster, who was a famous scientist in the 19^{th} century, argued that people with monocular vision (vision from one eye) could perceive depth just as well as people with normal stereo vision, and even believed that in some cases monocular viewing could be superior [19]. However, Julesz [84] dispelled such notions by proving that the stereoscopic depth cue alone was sufficient to induce a vivid depth sense. He demonstrated this fact by inducing binocular depth perception with computer generated random dot stereograms, lacking all depth cues except for disparity, i.e. small horizontal shifts (parallax) between corresponding points in the image pair;

Figure 1.2 shows an example of a random dot stereogram.

Figure 1.2: An example of a random dot stereogram (from [84]).



Numerous experiments have also shown that binocular vision can provide some significant advantages over monocular vision, for example it can aid in the following tasks: Relative depth judgements [72]; spatial localisation, i.e. the ability to concentrate on objects at certain depths while ignoring objects at other depths, thus aiding comprehension of large amounts of complex data [103]; breaking camouflage [164]; noise reduction [91] and improved detection thresholds for visual signals in noisy backgrounds [134] (also known as binocular unmasking); surface material perception from lighting effects such as lustre; and judgment of surface curvature [80].

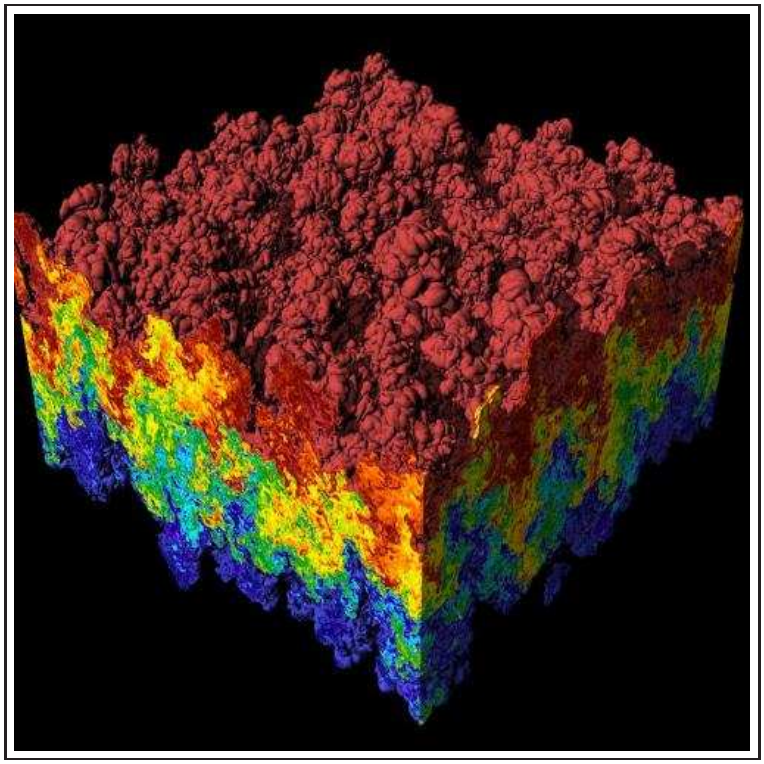
1.2 Stereoscopic scientific visualization

Scientific visualization is an interdisciplinary branch of science concerned with visualizing, comprehending and analysing three-dimensional phenomena, such as geological data sets and medical systems, by using concepts from computer graphics (see Figure 1.3 for an example of scientific visualization).

> Scientific visualization became an important field in the late 1980s, when scientists and engineers realized that they could not interpret the prodigious

quantities of data produced in supercomputer runs without summarizing the data and highlighting trends and phenomena in various kinds of graphical representations [47, chap. 1].

Figure 1.3: A scientific visualization of a simulation showing a RayleighTaylor instability caused by two mixing fluids, from [93].



An old adage goes that ‘a picture speaks a thousand words’, but a stereoscopic image can convey many more and it is now becoming apparent that the key to understanding today’s scientific data lies in stereoscopic visualization. Creating successful stereoscopic imagery requires a deep understanding of three fields of study: the human visual system and depth perception; three-dimensional display technologies; and computer graphics. Exploring each field is important in order to identify and bring together vital concepts needed to improve the efficiency and quality of generating stereoscopic images.

1.2.1 Human visual system and depth perception

While much experimentation and observations have been carried out regarding the human vision and depth perception [18,20,40,43,149,160] our understanding is still very limited; for example, there is still some debate on what type of model the human visual systems uses to combine different depth cues [45] and the possible benefit gain from each depth cue [16, 160]. This should not be that surprising considering how complex and powerful the human visual system is: some mechanisms that we know the visual system is responsible for, like solving the "correspondence problem" (matching corresponding points in each retinal image), still remain a challenge even for our most powerful supercomputers, yet most humans constantly solve this problem without even thinking about it.

Creating a stereoscopic image that is comfortable and avoids various depth distortion phenomena is challenging [83]. Improving our understanding of human depth perception is important if we wish to fully adopt 3D displays, and create comfortable and safe stereoscopic content; currently the long term health implications from viewing three-dimensional displays and low quality stereoscopic imagery is unknown. An important consideration for a good stereoscopic rendering algorithm would therefore be the inclusion of some type of depth control mechanism to aid in the creation of comfortable stereoscopic imagery. So far however, academics have either focused on developing algorithms for efficient stereoscopic rendering, or stereoscopic depth control, but not both in combination.

1.2.2 Three-dimensional display technologies

Three-dimensional display technology has come a long way since the invention of the Wheatstone stereoscope and there is now a bewildering range of technologies available, however, they can be very broadly distinguished between volumetric, holographic and planar surface displays (see [12, 101, 108, 115] for a more detailed review). Regardless of the type of technology used to present a stereoscopic image, they all require at least two images generated from slightly horizontally shifted cameras; in the case of holography and multiview displays, many hundreds of images may be simultaneously projected.

1.2.3 Computer graphics

Computer graphics is a broad field which includes the creation, storage, and manipulation of models and images of objects. The process of generating a digital image is known as rendering and usually involves mapping a 3D computer model onto a 2D projection using a virtual camera. Computer graphics algorithms have evolved rapidly since the 1980's and are capable of rendering almost photo-realistically, however, interactive graphics algorithms are usually restricted to only simple approximations of the behaviour of light.

Computer animation involves a single virtual camera and multiple renditions and transformations of the scene for each frame of a sequence. Generally each frame is rendered almost independently and there is very little calculation reuse. However, some limited research has been presented on identifying and eliminating redundant calculations for rendering stereoscopic images; these redundancies arise from the perspective coherence available between horizontally shifted cameras.

Computer graphics is mostly concerned with real-time and interactive graphics where the scene must be rendered and sent to the display at least 30 times per second in order to trick the mind into perceiving smooth animation. These requirements were originally very demanding on the computer system and lead to a solution known as z-buffer triangle rasterization; in order to reduce the number of calculations and required memory bandwidth, the scene is decomposed into individual primitives - almost always into triangles. The triangles can then be positioned, scaled and projected onto a 2D domain before finally being rasterized (decomposed further) into individual pixels ready to be displayed on the monitor.

Most computers have a specialized component known as the graphics processing unit (GPU), or graphics card, which is a dedicated piece of hardware for 3D graphics rendering computations. Modern GPU's are incredibly powerful, capable of crunching through billions of calculations per second, and exploiting their latest capabilities will likely be key to developing an efficient stereoscopic rendering algorithm.

A common problem in computer graphics is the aliasing phenomena, e.g. moire fringe patterns, which is caused by the discrete sampling of the scene during rendering. Although large amounts of effort have been dedicated to solving this problem for traditional 2D rendering [47, chap. 14], oddly very little attention has been given to anti-aliasing

when developing stereoscopic algorithms even though the effects of aliasing are compounded during stereoscopic viewing and can introduce various depth distortions [125].

1.3 The research problem and objectives

Research in computer graphics has in the past mostly been directed towards triangle based rendering algorithms, however, there are alternative primitive representations. A growing sub-field of computer graphics is point-based rendering [99, 132] where points are used to represent the scene instead of triangles. Points can be more efficient than triangles when the projected primitives are smaller than the pixels of the display screen and are a great benefit in highly detailed scenes. Another reason for the growing popularity of point-based rendering is that particles and point-cloud data sets are becoming the basic data unit found in a wide range of applications and research fields; for example, particles are used to represent the mass in the universe in cosmological numerical simulations [29] and the topology measured by airborne laser scanning [11].

Due to the size of many scientific point data sets, rendering even a single view can be expensive; therefore, the key to successfully visualizing point data sets on multiview displays is to limit the number of views to an acceptable minimum. Previous studies are based on subjective scores and suggest a high number of views are required, which if true limit the application of multiview displays. However, to date no research has been conducted on the affects of viewpoint density on depth perception. Using human factor trials, this thesis explores how many views may be required when task performance is taken into consideration rather than aesthetic qualities.

Most stereoscopic algorithms available today have been designed to work efficiently for either two views precisely or many views (100+) and are targeted at triangle based rendering. The little amount of research available for stereoscopic point based rendering focuses on recreating surfaces from the points; there has been no research presented on a stereoscopic algorithm for purely uncorrelated 3D point based representations.

Based on the results from the study a number of novel techniques are explored to stream line the stereoscopic rendering process for uncorrelated points. The research presented in this thesis is timely given that scientists are effectively drowning in point

based data due to the unprecedented resolution and accuracy of modern data acquisition tools [21] and ever increasing power of parallel supercomputers; this work will aid in scientific analysis and discovery of point data sets.

1.4 Thesis organisation and contributions

- Chapter 2 gives some general background information required for the understanding of the material in this thesis and explores, in greater detail, the scientific fields identified above: the human visual system and depth perception; three-dimensional display technologies; and computer graphics.

- Chapter 3 discusses a human factors trial designed to measure task performance of subjects using multiview displays with varying amounts of depth and viewpoint densities with the goal of quantifying the optimum number of views for a 3D display. Contributions from this study include:

  - An in depth evaluation of the requirements of multiview displays.
  - The design of a display apparatus for simulating multiview autostereoscopic displays of varying viewpoint density.
  - A path tracing task, based on [160], to evaluate human 3D task performance on multiview displays.
  - A recommendation for multiview display system designers that low viewpoint densities may be sufficient to enable effective path searching task performance when occlusion is not an overriding factor.
  - Results showing that binocular stereo and motion parallax do not always have an additive effect on depth perception (as previously suggested by a number of studies [160, 162]). We show for a similar task but with limited occlusion that the stereo cue dominates over the head motion parallax cue.
  - Confirmation of previous results [140] that low magnitudes of stereoscopic depth are useful to provide a task benefit.

- Chapter 4 explores the potential for increased efficiencies when rendering points for low viewpoint densities as concluded in the human factors trial. The chapter describes a theoretical multiview algorithm for efficiently rendering uncorrelated 3D points by taking advantage of the perspective coherence available in the views as well sharing the lighting calculations. Set-up costs are kept to a minimum so that performance gains can still be realised for displays with a low viewpoint density as recommended in chapter 3. A key strategy for the algorithm which is unique compared to other stereoscopic algorithms is that it assumes all the particles are spherical and translucent which allows a number of stages in the rendering pipeline to be eliminated. Also the use of additive blending removes the requirement for depth sorting and occlusion handling thus greatly improving efficiency at the reprojection stage.

- Chapter 5 shows that the number of reprojection calculations can be reduced further (at the cost of some accuracy) by taking advantage of the display's limited amount of stereoscopic resolution. This novel approach involves dividing the scene into slices and repojecting each slice instead of each point. Some of the latest programmable shader capabilities are utilised to implement the algorithm in a two-stage rendering pass. Aside from the performance improvements, another benefit of the algorithm is the opportunity for much greater control of the stereoscopic depth at little or no extra cost.

- Chapter 6 describes some further applications and extensions possible for the algorithm described in the preceding chapter.

- Finally chapter 7 summarises the main results of this thesis and discusses areas for further investigation.

Chapter 2

Background and previous work

This chapter describes the general background information required for understanding the material in this rest of the thesis. The first section discusses the basics of human depth perception and the differences between viewing natural content and stereoscopic images. The second section looks at the advantages and disadvantages of different stereoscopic display technologies, paying particular attention to multiview displays. The final section discusses 3D image generation, including camera models, depth control, stereo-aliasing and multiview rendering algorithms.

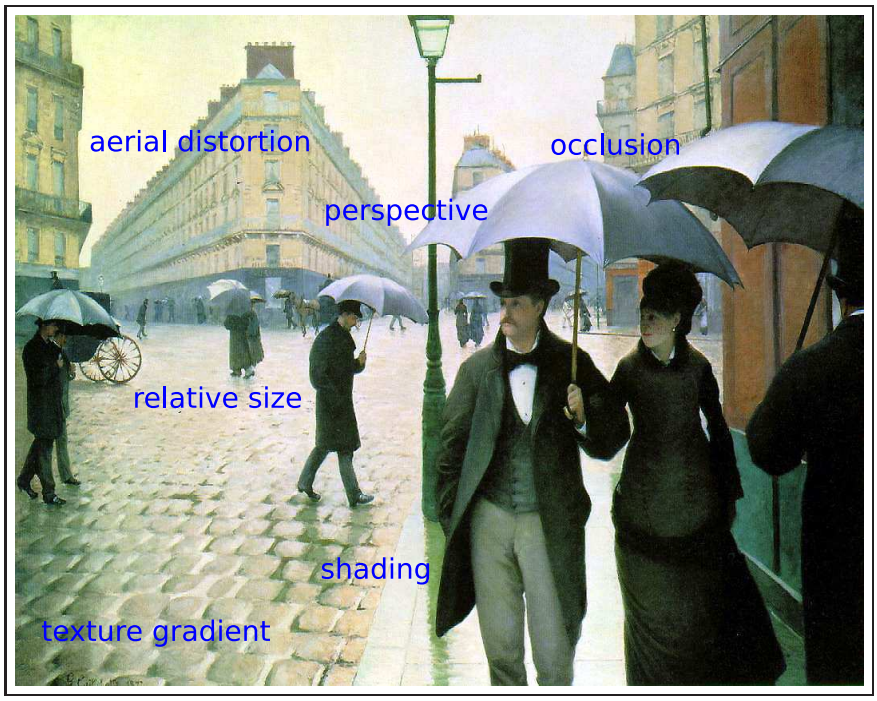
2.1 The basics of human depth depth perception

The world we live in is three-dimensional and for most of us perceived as such: for example, we are aware of characteristics such as distances (location), depth, shape, size and orientation. However, when light enters our eyes, it falls onto a two-dimensional surface known as the retina; the three-dimensional structure of our environment must be teased out from the flattened retinal images using various psychological and physiological depth cues.

Psychological depth cues (also known as pictorial depth cues) include: linear perspective, lighting and shadows, aerial perspective, relative size, interposition or occlusion, texture gradient and colour [108]. Psychological cues are considered monocular because they can be observed with a single eye and can give an impression of depth even in a flat two-dimensional image such as a photograph or painting as illustrated in Figure 2.1.

Indeed, artists have known about monocular depth cues since the renaissance period and have used them to great effect.

Figure 2.1: Paris Street; Rainy Day, 1877 by Gustave Caillebotte shows how effective pictorial depth cues can be in creating the illusion of depth even from a flat surface (closing one eye also helps), from [23].



The four known visual physiological cues [1] available to humans are: accommodation, vergence, motion parallax and binocular disparity.

Accommodation and vergence are categorised as oculomotor depth cues because they are derived by feedback from differences in the muscular tension in the eyes. Accommodation is the action of contracting or relaxing the ciliary muscles so as to change the shape and optical power of the lens, focusing incoming light rays onto the retina so as to form a clear image. Blur information from different states of accommodation can also provide a cue to relative and absolute depth [106]. Binocular vergence is the rotation of the eyes,

[1] There are also non-visual physiological cues to depth, such as sound, balance from the inner-ear, haptic cues such as tactile and kinesthetic; however, they are beyond the scope of this thesis (see [15, chap. 3] for more details).

either convergence or divergence, towards a point of interest so that it can be fused into a single image (more on this later).

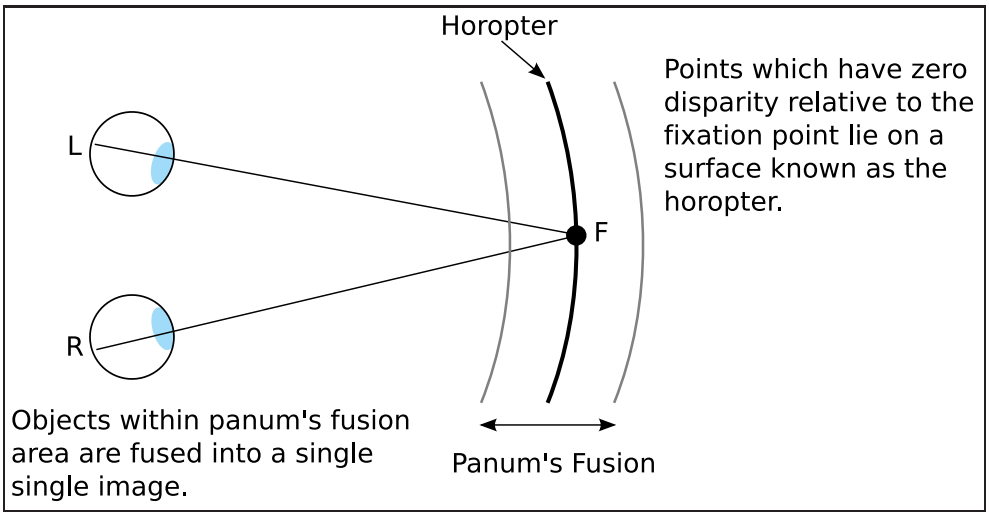
Motion parallax occurs when either the observer or scene is in motion: objects further away will move across the retinae more slowly than objects which are closer to the viewer, which allows relative depth judgments to be made; its effects are readily noticeable in the car, where objects in the far distance appear stationary and objects close by rapidly travel across the observer's field of view.

Binocular disparity refers to the difference between the image formed on the left and right fovea. If the images formed on the retinae were somehow superimposed and printed as a photograph, two horizontally displaced but overlapping images would be seen. The small differences between the retinal images allow the brain to perceive depth [30]; this process is called stereopsis (literally solid seeing). Stereopsis is only available to animals with two forward facing eyes.

Figure 2.2 illustrates stereoscopic depth perception under natural viewing conditions. The eyes rotate towards a fixation target, adjusting their accommodation state and bringing the region into focus. The fixation point is projected onto the exact same position for both the left and right retinae, whereas points in front or behind it will project to different locations: stereoscopic depth judgments are therefore relative to the fixation target. Points extending from the fixation target that have zero binocular disparity, and therefore perceived to be at the same distance from the observer, form the horopter [31]. The Vieth-Müller circle, which is shown in Figure 2.2, represents the theoretical horopter, however in reality the horopter is known to be a complex shape and to have non-linear characteristics [13, 53].

Along the horopter is also a volume known as Panum's fusional area; points lying within this area are fused by the visual system and will be seen singularly with good depth perception. However, objects outside this area cannot be fused and are actually perceived as double vision. This phenomena is known as diplopia and occurs all the time in natural viewing. A simple experiment described in [102, chap. 2] can be carried out by the reader to confirm the presence of diplopia: by holding a thumb out at arms length and focusing on it, the observer should notice that objects in the background will appear twice; if the observer then focuses on the background, it will appear as a single image and

Figure 2.2: Geometry of binocular vision.



the thumb will appear twice.

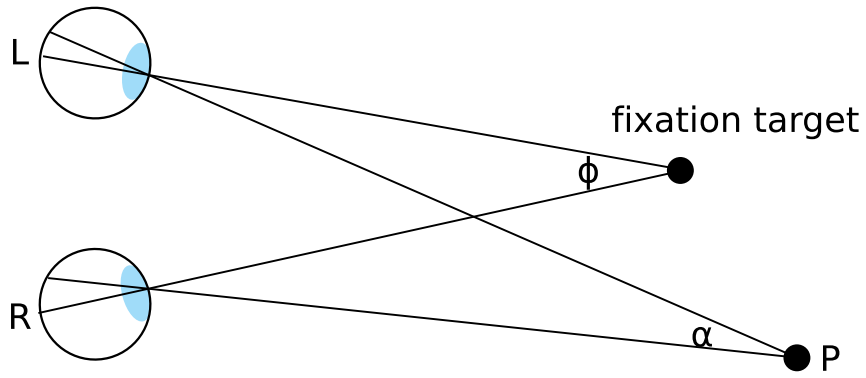
The limits of Panum's fusional area is elliptic and allows for greater horizontal disparities to be fused than vertical disparities. The area also increasingly expands towards the periphery of our vision, which is one reason why diplopia is not really noticed. Another reason is that the limits of fusion are usually close to the limits of the depth of field [135] and therefore objects further away appear increasingly blurred rather than double. Blur also helps maintain a greater fusional range [154]. The Panum's fusional area varies greatly from person to person and is affected by many factors such as spatial and temporal properties of the stimulus [33, 154].

The human depth perception is amazingly sensitive; a comparison experiment between two rods at slightly different depths revealed subjects were capable of detecting differences in depth of as little as 2 sec arc with 75% accuracy [72]. Tyler [154] argues binocular disparity allows a healthy human to perceive depth differences of as little as one-thousandth of an inch for fixation distances of 10 inches away, and at distances of up to 2 miles away can detect whether an object is closer than the horizon.

2.1.1 Measuring stereoscopic depth

There are numerous methods for measuring stereoscopic depth in both real and virtual environments. The most common methods are briefly described in this section since a basic understanding is vital in order to compare results from stereoscopic studies fairly.

Figure 2.3: Definition of retinal disparity.



Retinal disparity

Retinal disparity in relation to an object of interest can be defined as the difference between the convergence angle of that object and the convergence angle associated with the fixation target: in Figure 2.3, the retinal disparity for P is $\theta = \phi - \alpha$.

Parallax

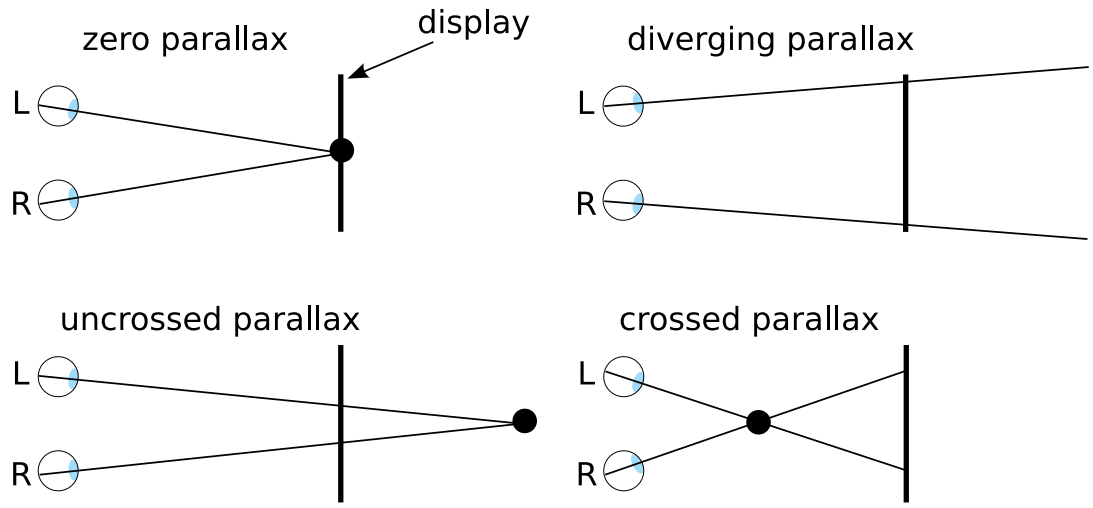
Stereoscopic displays produce parallax, which are differences between homologous points or pixels in the left and right images on the screen; this in turn produces retinal disparity in the eyes when viewed correctly. There are four types of horizontal parallax which induce the stereoscopic depth cue (see Figure 2.4):

- Zero parallax occurs when the homologous points in the two images are at identical positions and thus the point is perceived to lie at the display.
- Uncrossed or positive parallax induces depth behind the display. When the lines of sight from the eyes to the image points are parallel, the object is perceived to lie at an infinite distance from the observer.

- Diverging parallax is an extreme form of positive parallax and occurs when the parallax is greater than the observer's eye separation. This phenomenon does not normally occur under natural viewing conditions, however if a viewer is capable of diverging his or her eyes and can successfully fuse the image, the object would be perceived at infinity behind the viewer [139].

- Crossed or negative parallax induces depth in the volume between the display and the observer.

Figure 2.4: Four types of horizontal parallax: zero parallax; uncrossed or positive parallax; diverging parallax; crossed or negative parallax.

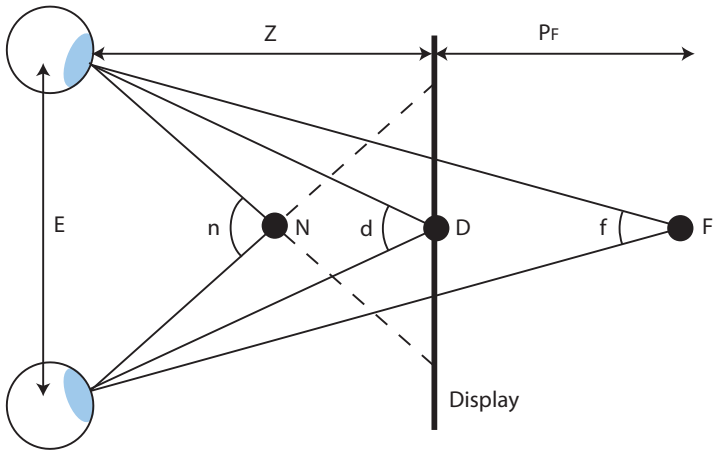


A problem with simply reporting the amount parallax in a scene is that the actual amount of depth perceived is unknown unless the viewing distance is also reported (eye separation is often assumed to average 6.5 cm). Scientists therefore, often prefer to report stereoscopic depth using angular disparities instead, with the intent of normalising the results by the viewing distance. However, there is still a problem with approach as explained below.

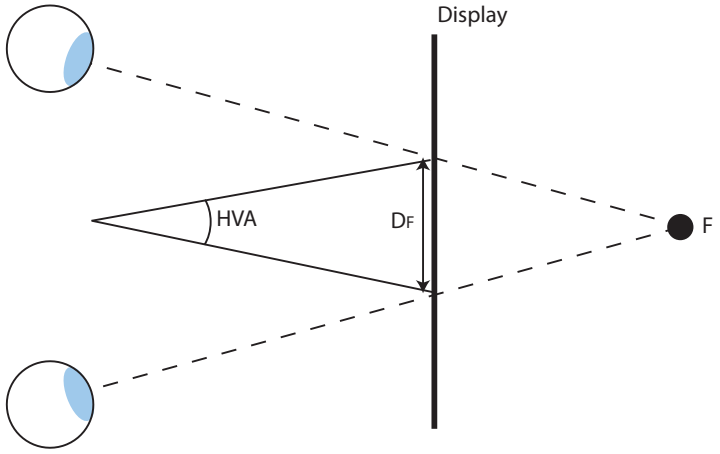
Angular disparity

The angular disparity, which is also known as the vergence difference, of a virtual point in depth is defined as the difference in angle between the eye vergence at the virtual point and

Figure 2.5: Angular disparity and horizontal visual angle (HVA) measurements in a virtual environment



(a) Angular Disparity



(b) Horizontal visual angle (HVA)

the eye vergence at the display screen. Referring to Figure 2.5(a), the angular disparity, F (positive disparity), can be calculated as follows:

$$\theta = d - f \tag{2.1.1}$$

and the angular disparity of N (negative disparity) can be calculated similarly by:

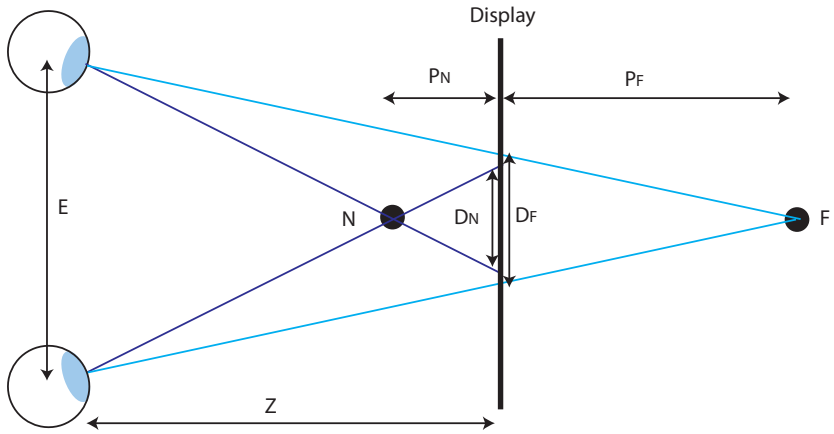
$$\theta = d - n \tag{2.1.2}$$

Angular disparities reflect retinal disparities if the observer is assumed to be fixating on the display screen.

Horizontal visual angle

Another similar measurement to angular disparity is to measure the horizontal visual angle (HVA) as shown in Figure 2.5(b). This is defined as the angle between the two corresponding points on the display screen from the centre of the eyes. Although HVA and angular disparity are calculated differently, they are in fact equivalent.

Figure 2.6: Geometric model of perceived depth for two points in front and behind a stereoscopic display.



Geometric perceived depth

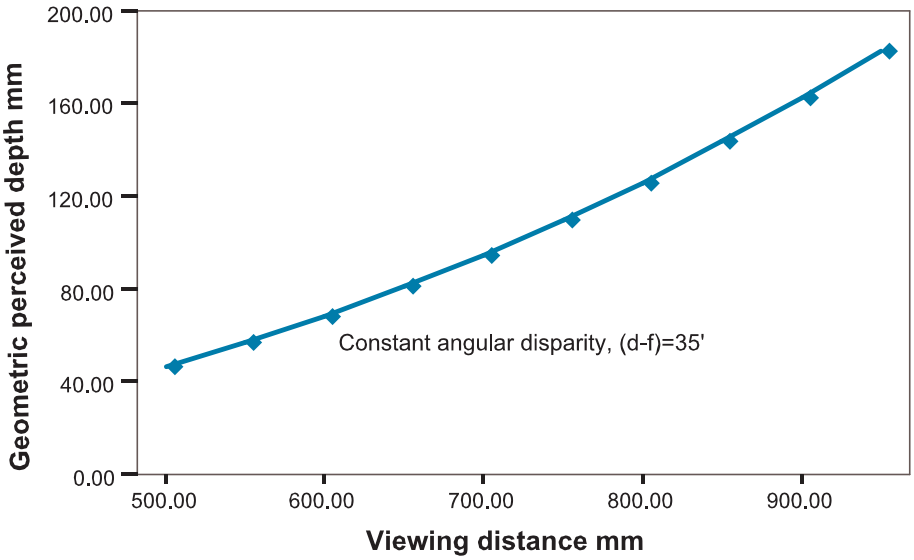
An alternative method is to attempt to measure the perceived depth using a geometric model. This assumes the human perception will recover Euclidean geometry and the observer is presented with an ideal display. In fact the geometric perceived depth (GPD) will be measured which is likely to be different from the true perceived depth. However, GPD models help us to understand the human depth perception using stereoscopic displays by identifying key geometric variables and their relationship with depth perception. A simple model described by [66] is illustrated in Figure 2.6. The eyes are separated by the interocular distance, E, and the viewer is positioned centrally to the stereoscopic display at a viewing distance, Z. The GPD for a virtual point behind the display with a positive pixel disparity, D_F, is calculated as follows:

$$P_F = \frac{Z}{(\frac{E}{D_F}) - 1} \tag{2.1.3}$$

Whereas, the GPD for virtual points in front of display with negative disparity (crossed disparity), D_N, is calculated by:

$$P_N = \frac{Z}{(\frac{E}{D_N})+1} \tag{2.1.4}$$

Figure 2.7: The geometric perceived depth still varies over a range of viewing distances when the angular disparity is kept constant [83].



Equations 2.1.3 and 2.1.4 identify a key property of stereoscopic displays: GPD is directly proportional to the viewing distance. However, as figure 2.7 demonstrates, a constant angular disparity does not maintain a constant GPD over a range of viewing distances [83]. This implies angular disparities from different investigations cannot be compared directly since the perceived depth will likely be different. Therefore, we believe GPD a more appropriate measure for informing stereoscopic depth quantities.

In order to compare angular disparities between different investigations, we can convert them into GPD values if the viewing distances are known. Referring to Figure 2.5(a), first the angle, d, must be calculated as follows:

$$d = 2\arctan(\frac{E}{2Z}) \tag{2.1.5}$$

Then the angle, f can be obtained from the given angular disparity, θ, by:

$$f = d - \theta \tag{2.1.6}$$

Finally the perceived depth can be calculated by:

$$P_F = \frac{E}{2\tan(1/2f)} - Z \tag{2.1.7}$$

2.1.2 Accommodation, vergence and depth of field

When a human looks at a point in space his eyes will automatically accommodate and converge together at that point. Vergence (rotation of the eyes) is triggered primarily by disparity [148], whereas accommodation is driven primarily by blur [27]; however, accommodation can also be affected by depth sensation from monocular cues [148]. Under natural viewing conditions accommodation and vergence are closely linked by reflex so that a change in state of either accommodation or vergence will trigger a change in the other automatically. Pupil size is also linked by reflex to accommodation and vergence and forms a complex feedback mechanism which is known as the near triad system [73, chap. 9]. The exact nature of the near triad system is still not fully understood; difficulties in investigating the human visual system can be partly blamed on the lack of non-intrusive measurement devices, and the use of trained subjects under non-natural viewing conditions which can significantly affect the results [70].

Given a point in space which the eyes are focused on, there is a depth range in which everything inside of it will appear sharply in focus, i.e. blur cannot be detected. This is considered to be the depth of field (DOF) and occurs because the eye is not a perfect optical system and the retina is not infinitely sensitive to optical blur. The DOF is affected by many factors, including pupil diameter, visual acuity, axial length of the eye, chromatic and spherical aberration and the the stimulus itself [54].

Two properties which 3D displays do not usually offer are accommodation cues and a natural DOF effect. Therefore, the eyes of the observer will usually have to be accommodated near the display screen to sustain a sharp image regardless of vergence. This causes a conflict to the accommodation vergence reflex. Evidence for this breakdown is provided by a study [148] that measured the accommodation and vergence, simultaneously, of a number of subjects using a stereoscopic display. The study shows that as an

observer fuses a 3D image there is an initial overshoot of accommodation which then recedes considerably while vergence remains constant; the same accommodation overshoot was observed in natural viewing conditions, however it was much smaller than during the artificial stereoscopic condition.

2.1.3 Viewer discomfort in 3D displays

It is well known that the disparity on a 3D display must be limited to maintain a comfortable viewing experience [83, 155, 167, 176]. There are a number of differences in viewing the stereoscopic display compared to natural viewing which are most likely responsible for the cause of discomfort.

Viewing discomfort and visual fatigue are often used interchangeably, however there is a subtle difference: viewer discomfort can only be measured subjectively, whereas visual fatigue is a decrease in visual ability and can be measured to some degree. Lambooij et al. [95] argue that subjective indicators, for example questionnaires, are not sensitive enough to measure visual discomfort in a reliable and accurate manner and should at least be combined with visual fatigue measures. The following measures can indicate the amount of fatigue: pupillary diameter and reactions; critical fusion frequency; visual acuity; near point refractionability; visual field; stereo acuity; fixation stability; accommodative response; magnitude of accommodation vergence crosslink-interaction (AC/A ratio); heterophoria; convergent eye movement; spatial contrast sensitivity; colour vision; light sense; blink rate; tear film breaking time; pulse rate; and respiration time [95].

The most often cited problem causing viewer discomfort is the accommodation vergence breakdown [79] as described above. Other problems are due to imperfections in the binocular image pairs, for example optical misalignments and imperfect image filters (see [92]). Optical misalignments can cause spatial distortions such as shifts, magnification, rotation and keystone. Imperfect filters can cause photometric asymmetries such as luminance, colour, contrast and crosstalk.

Crosstalk is unfortunately present in nearly all auto-stereoscopic displays, and humans are extremely sensitive to it: crosstalk as little as 1×10^{-4}% can be detected. Building an auto-stereoscopic display with no perceptible crosstalk is extremely challenging. However, Huang et al [75] suggest that 0.1% crosstalk is acceptable to most people for most

types of stimulus and therefore 0.1% can be considered a reasonable target.

Analysing comfortable depth ranges

Many factors affect the fusional ranges on stereoscopic displays. Improving viewing comfort by reducing crosstalk and binocular imperfections are two effective methods. However, fusional ranges are also affected by spatial and temporal frequencies of the stimulus; field of view; the surrounding environment; and DOF [79, 173].

Most investigations measure the maximum depth that can be fused before diplopia is perceived, rather than measuring subjective comfort ratings. A classic paper is Yeh and Silverstein's investigation [175] which reported fusion limits of up to 1.57 deg for uncrossed disparity and 4.93 deg for crossed disparity when the stimulus was presented for 2 sec. Some other investigations reported the following fusion limits: approximately 4 deg uncrossed, 3 deg crossed [79]; with a viewing distance of 70 cm, fusible depth limits for a simple scene were typically greater than 3.9 deg uncrossed and 4 deg crossed, however with a complicated scene fusible depth limits drastically decreased to 50 arc min uncrossed and 53 arc min crossed [83].

Subjective studies on the other hand, usually report that much lower disparity values are required for comfortable viewing. For example one study [171] showed that only about 35 arc min of disparity was acceptable when a sharp background was present in the stereoscopic image. The study also showed that disparities could be increased without complaint as the background became blurred.

Lambooij et al. [95] review of the literature on viewing comfort recommends 1 degree of disparity as a general rule-of-thumb. However, there are number of problems with generalising the results from the literature. The most obvious is that comfortable depth ranges will almost certainly be much less than fusible depth ranges. Take the colour anaglyph for example, even with small disparities which are easy to fuse, observers often get a head ache after a while due to binocular rivalry. It would seem probable that stereo depth should be at least limited to the DOF so as to minimise vergence accommodation breakdown. An investigation by Yano et al. [173] which measured accommodation before and after stereo viewing and evaluated the visual fatigue with subjective responses, indicated that within the DOF visual fatigue was comparable to watching the scene with

out stereo depth, however visual discomfort was clearly experienced when the perceived depth exceeded the DOF. Interestingly, visual fatigue was experienced even for depth ranges within the DOF when motion was present in the stimulus. The DOF was assumed to be ± 0.2 D which equates to ± 0.82 deg or 133 cm behind the screen and 87 cm in front of the screen at a viewing distance of 105 cm. This amount of depth is still quite large and while it may be fusible for a still image, evidence from [175] suggests when the stimulus presentation duration is short (less than 200 ms) fusion limits are drastically reduced to 27 min arc for crossed disparity and 24 min arc for uncrossed disparity. It is plausible that diplopia would have been experienced which would of course cause some discomfort. Further evidence for visual discomfort within DOF depth limits can be found in [174]. These results are significant because they imply that either the accommodation vergence mismatch is not the most important problem to solve for stereoscopic displays or that our current understanding of the near triad mechanism is inadequate.

Ultimately comfortable depth limit recommendations vary greatly due to the differences in experimental set-up, stimulus and display characteristics. For this reason it is difficult to estimate the comfortable depth limits for a given display. It is also important for any investigation which analyses an effect due to stereopsis from a stereoscopic display to keep the maximum disparity within comfortable limits; otherwise, comfort issues may likely have an adverse influence on the results. A sensible option is to keep depth limits within the most conservative recommendations to ensure minimal discomfort regardless of the stimulus. For desktop viewing conditions, i.e. approximately 19 inch screen and a viewing distance of 70 cm, we believe, from experience, that the recommendation of ± 10 cm by Jones et al. [83] to be reasonably good.

2.2 3D display technologies

While there are numerous stereoscopic display technologies, they can be broadly classified as volumetric, holographic or planar surface displays. A brief introduction of these technologies is presented (for more information see [12, 101, 108, 115]) before moving on to look at auto-stereoscopic planar surface displays in more detail.

- Stereoscopic planar surface displays attempt to reproduce depth by displaying two

or more flat images. Many of these displays require viewing aids in order to filter out the appropriate view for each eye, for example, polarising crystal shutter glasses or colour anaglyphs. However, it is also possible to create viewing windows in space by using optical devices, such as lenticular lenses or parallax raster barriers, which are built into the screen so that glasses are not needed; this method of viewing is known as auto-stereoscopic.

Some auto-stereoscopic displays can present the viewer with more than two different perspective images and are known as multi-view displays. These additional views provide a wider viewing angle for the display without the need for head-tracking, allow a 'look-around' effect, similar to the experience when looking out of a window and moving laterally, and can easily support multiple observers. The look-around effect enables the perception of head movement induced motion parallax and there is evidence that the combination of this cue with stereo viewing greatly improves depth perception [160]. Commercially available multiview displays have in the order of ten simultaneous views [127], typically repeating as a block around the display, while research projects have demonstrated displays with over a hundred views [113].

Although stereo planar surface displays can provide binocular disparity and vergence cues as well as motion parallax with either head-tracking or multi-view technology, correct accommodation or focusing cues cannot easily be reproduced.

- Holographic displays are able to reconstruct the exact light wavefronts reflected off any object and can provide all the visual cues including accommodation. Holograms have the potential to be virtually indistinguishable from real life scenes. A hologram stores the interference fringe pattern formed when a coherent light source is reflected off any object. When the holographic film is illuminated with the same coherent light source at the same angle as recording, the film acts as a diffractive lens which will reconstruct the original light waves.

  Holography suffers major drawbacks in that the slightest movement to either the recording devices or scene during recording will ruin the hologram. Also to provide correct accommodation cues the interference grating needs to be recorded at a

very high resolution; a typical holographic display has a spatial frequency exceeding 1500 lines per millimetre [59]. Other problems in implementing holographic displays due to the limitations imposed by the nature of light and diffraction, include the difficulty of achieving holograms larger than a few centimetres, speckle and modulation noise, narrow fields of view and the requirement of huge computation to calculate the holograms at interactive refreshment rates [10]. For these reasons holography remains a challenging field.

A common way to ameliorate the bandwidth problems is to eliminate vertical parallax, which does not contribute to the depth sense, and only provide horizontal parallax. The required bandwidth can be reduced further by approximating the continuous parallax of holograms with multiple discrete perspective views dense enough so that they appear to provide a continuous range of perspective; these are known as holographic stereograms. However, holographic stereograms have their own set of problems, including inaccurate accommodation and inter-perspective aliasing due to insufficient sampling of the wavefronts [58].

- Volumetric displays are unique from other 3D displays in that they don't simulate depth but actually reproduce it by illuminating well defined regions in physical space. Volumetric displays are a promising solution to 3D viewing because they can reproduce all the depth cues including accommodation without the requirement of visual aides. While there are many different types of volumetric displays, they can be broadly distinguished into three categories: swept-volume or swept-surface; solid or static; and slice stacked.

  Swept-solid displays project light onto a moving surface designed so that it will eventually fill the entire volume. A single 3D image is perceived because the human persistence of vision fuses the time-series of regions. A typical example of this display is the Actuality Systems Perspecta Volumetric 3D Display which projects 2 x 198 images onto a rotating disk at 900 rpm [26].

  Solid volumetric displays do not use any moving parts, but instead use an emissive element at each voxel or 3D pixel in the scene volume. Each voxel must be transparent when switched off but opaque or luminous in the on state. One example of

this type of display uses two lasers perpendicular to each to other to excite certain regions in a volume filled with gas [100].

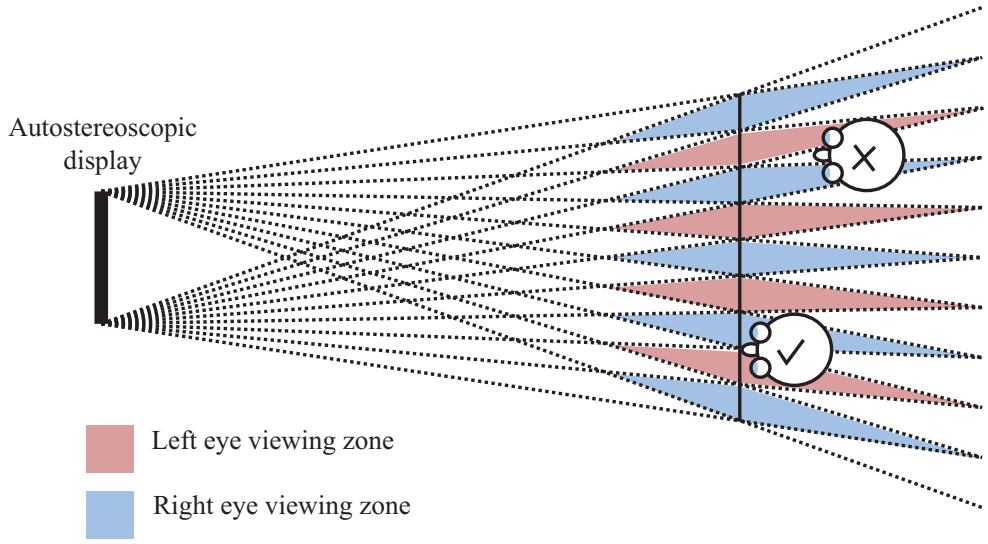
Slice stacked displays are considered more of a hybrid technology since they use multiple displays at different depths. The first known example of this display was described by Louis Lumiere in 1920 and involved stacking successive optical tomography photographs which created a sort of photosculpture. More recently, Akeley [7] built a volumetric display which presents three different depth planes to the viewer and works by using six beam-splitters and a high resolution LCD display.

A common misconception with volumetric displays is that they can not reproduce viewpoint dependent lighting effects such as occlusion. However, this is only true if the reconstructed voxels are translucent and isotropically emissive, for example the Perspecta Volumetric 3D Display which uses a highly diffuse rotating screen. Cossairt et al. [28] argue that replacing the diffuse screen common in time-sequential volumetric displays with one that controls the direction of light such as a translucent screen with unidirectional diffusion or a field of micro-lenses, can result in viewpoint dependent voxel reconstruction.

Like holographic displays, volumetric displays have serious bandwidth problems due to large number of views required and therefore are very expensive. Another shortcoming is that current implementations exhibit a large footprint since the depth is physical rather than simulated; although theoretically, time sequential displays can be built to project imagery outside of the volume swept by the rotating screen [28].

While volumetric and holographic displays are still actively being researched, the difficulties associated with these technologies are currently limiting their wide use and applicability. We believe auto-stereoscopic displays offer the best advantages, no glasses, wide viewing freedom, relatively low cost, and are rapidly growing in popularity; therefore, the rest of this thesis focuses on auto-stereoscopic displays, especially multiview.

Figure 2.8: Many non head-tracked two-view auto-stereoscopic displays repeat the two different perspective views across all the viewing windows. This means even at the ideal viewing distance an observer has a 50% chance of being positioned incorrectly and perceiving a pseudoscopic image. This diagram has been adapted from [38].



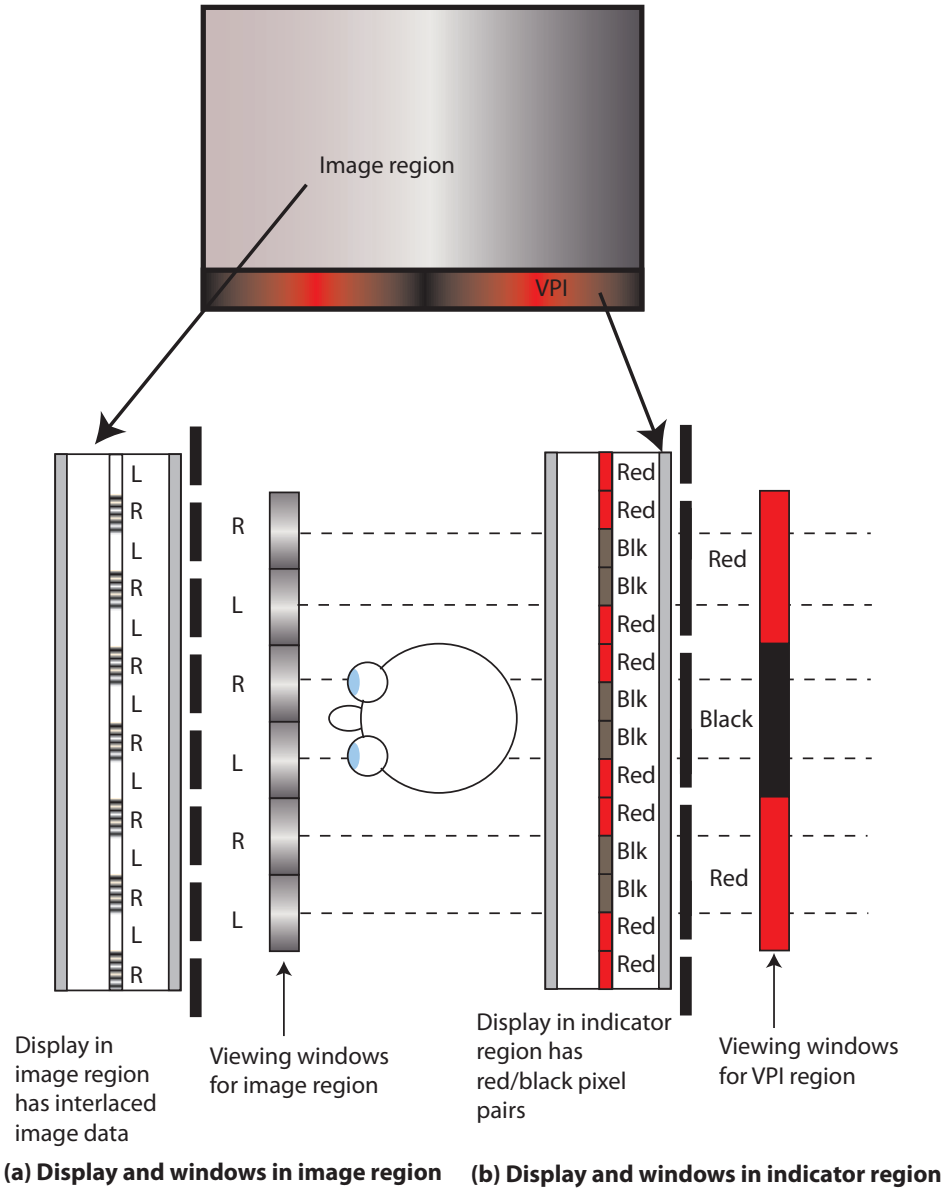
2.3 Planar surface auto-stereoscopic displays

2.3.1 Two-view auto-stereoscopic displays

Some of the earliest auto-stereoscopic display were built using either parallax raster barriers [78] or sheets of lenticular lenses [63]. In a two-view display the optical device separates two different images into repeating and alternating viewing windows across the entire viewing volume as illustrated in Figure 2.8. While simple and relatively cheap, there are a number of problems with displaying only two different images. A significant problem, common with all types of planar stereoscopic displays, is the shearing or false rotation phenomenon: as an observer moves laterally objects perceived in front of the screen appear to shear in the same direction as the observer, whereas objects behind the screen shear in the opposite direction; this leads to an unnatural distortion, which is exasperated with larger amounts of parallax and perceived depth [119]. Another problem, unique to auto-stereoscopic displays, is that even at the ideal viewing distance there is only a 50% chance that the observer will be positioned correctly (see Figure 2.8). Sitting

in an incorrect position leads to a pseudoscopic image [38] (inverted depth perception) which is not easily apparent to novice viewers.

Figure 2.9: The VPI display is composed of a image region and an indicator strip at the bottom of the display which helps the observer find the correct viewing position. This diagram has been adapted from [66].



Parallax barrier displays can use a viewing position indicator (VPI) as described in [57, 168] to aid correct positioning of the observers. The VPI takes up a small strip of pixels at the bottom of the display and is composed of a pattern of red and black stripes

with the pitch of the parallax barrier set to twice that of the rest of the display. When the observer is in the central viewing position and at the ideal viewing distance, the indicator appears completely black across the entire width. As the observer moves away in any direction from the ideal viewing position, the indicator will appear increasingly more red in colour (see Figure 2.9). The VPI is not perfect since it can be red in some of the correct orthoscopic viewing positions; however, Holliman [66] argues this is a reasonable trade-off for guaranteeing an orthoscopic image with the best image quality possible for the display when the VPI is seen as black.

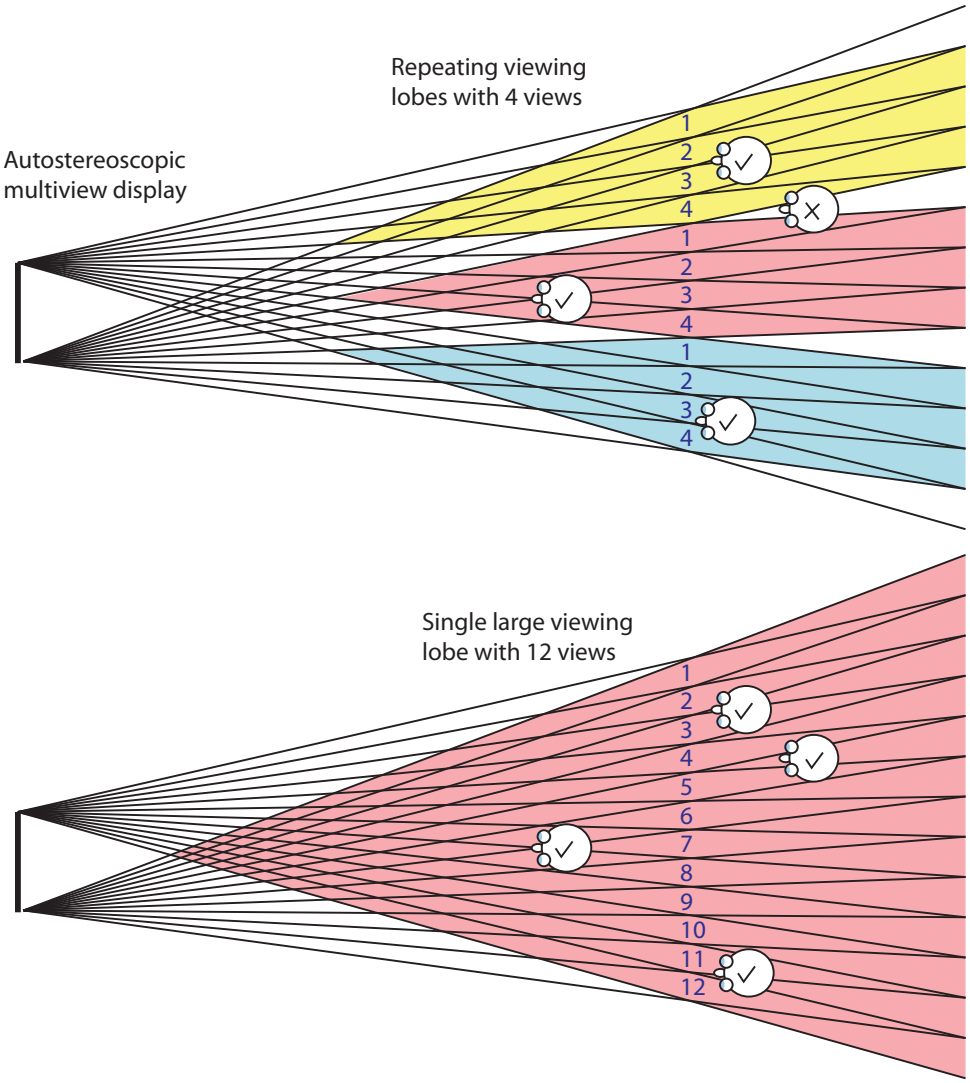
Instead of sacrificing pixels for a VPI, the display can be mounted on a rotating platform coupled with head-tracking technology so that the observer and display are always "face-to-face", as demonstrated by the Heinrich-Hertz-Institut [103]. This approach gives excellent viewing freedom without any spatial distortions in the scene; however, the mechanical movement device must be very robust so as to avoid failure and fast enough to keep up with the movement of the observer without any noticeable lag, which can considerably add to the cost of the display. Another display which uses head-tracking was built by NYU and avoids mechanical steering by using a LCD electronically programmable parallax barrier [121–123]: by varying the pitch and aperture of the transparent slits, the viewing windows can be steered to the same position as the observer's eyes. Programmable parallax barriers also enable the tracking and support of more than one viewer simultaneously [124].

2.3.2 Multiview auto-stereoscopic displays

An alternative solution to increasing viewing freedom is to show more images in the viewing windows i.e. a multiview display. Multiview displays can be set-up in one of two ways: a single large viewing lobe with many different perspective views can be presented, which supports the 'look-around' effect (see Figure 2.11(b)); or fewer views can used but repeated across multiple viewing lobes (see Figure 2.10).

Providing a single large viewing lobe with a dense number of views emulates a more natural viewing experience: in a natural scene, light waves propagate from every point lit up by a light source, as shown in Figure 2.11(a), presenting the observer with potentially an infinite number of different perspective views; simulating the full light field in this

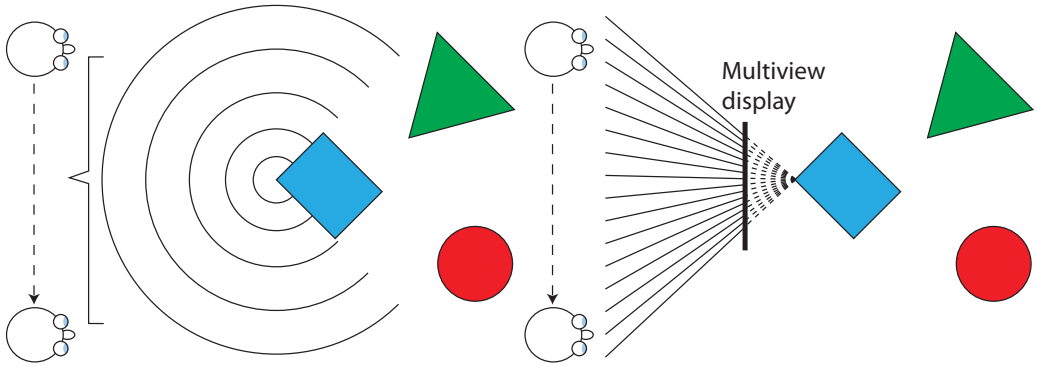
Figure 2.10: Multiview displays can provide a single viewing lobe with many views or limit the number of views and repeat them across multiple viewing lobes. An observer within any of these viewing lobes will perceive a stereoscopic image.



way, guarantees orthoscopic viewing for all viewers and can reduce or eliminate the false rotation phenomenon. However, there are serious technological challenges to building multiview displays capable of producing many high quality images.

Apart from increased viewing freedom, another less known advantage of multiview displays is their ability to support observers with varying eye separations while theoretically providing the same amount of GPD if there are multiple views within the average interocular separation (see Figure 2.12). If there are not enough different perspective

Figure 2.11: Light waves reflecting off a single point in a natural scene propagate and present an infinite number of different perspective views to the observer. Multiview displays approximate this natural way of viewing by providing a discrete number of different perspective viewpoints.



(a) Natural scene (b) auto-stereoscopic multiview display

views, a problem known as the flipping effect can become apparent as the observer's eyes moves from one viewing window into another (see Figure 2.13). A number of past studies have investigated this phenomena and observers' sensitivity to it in relation to the viewpoint density of the display; these studies are discussed in section 2.3.2.

There are a number of different methods for creating viewing windows; Dodgson [36] broadly distinguishes between three types of multiview technologies:

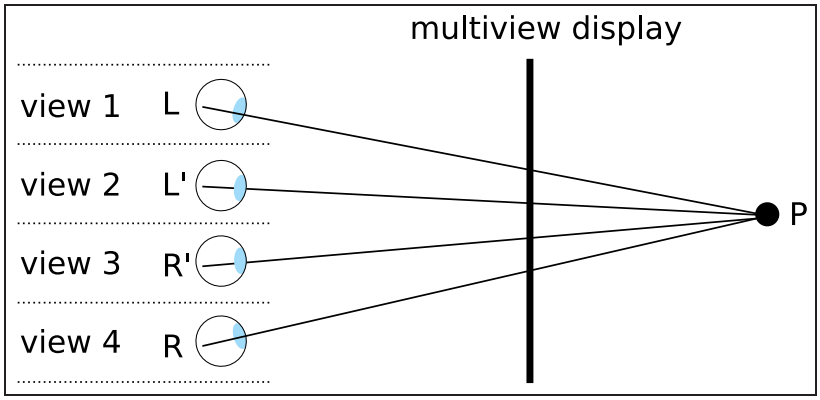
- Spatial multiplexing - The available resolution of the display is shared across the different viewing windows.
- Multi-projector - A separate display is used for each view.
- Time-sequential - Different images are presented sequentially to each viewing window using a display with a very fast refresh rate.

Each type of multiview technology will be discussed in further detail and display examples given.

Parallax raster barrier design

The optical elements used in many spatial multiplexing auto-stereoscopic displays are based on either parallax raster barriers or lenticular lens sheets. Even though the principles

Figure 2.12: The observer with left eye L' and right eye R' has a smaller eye separation than the observer with left eye L and right eye R, but both observers perceive the same amount of stereoscopic depth.



of parallax barriers and lenticular sheets have been known for over a century, constructing a display with a sufficiently precise pixel pitch has only been achievable relatively recently with LCD technology [36].

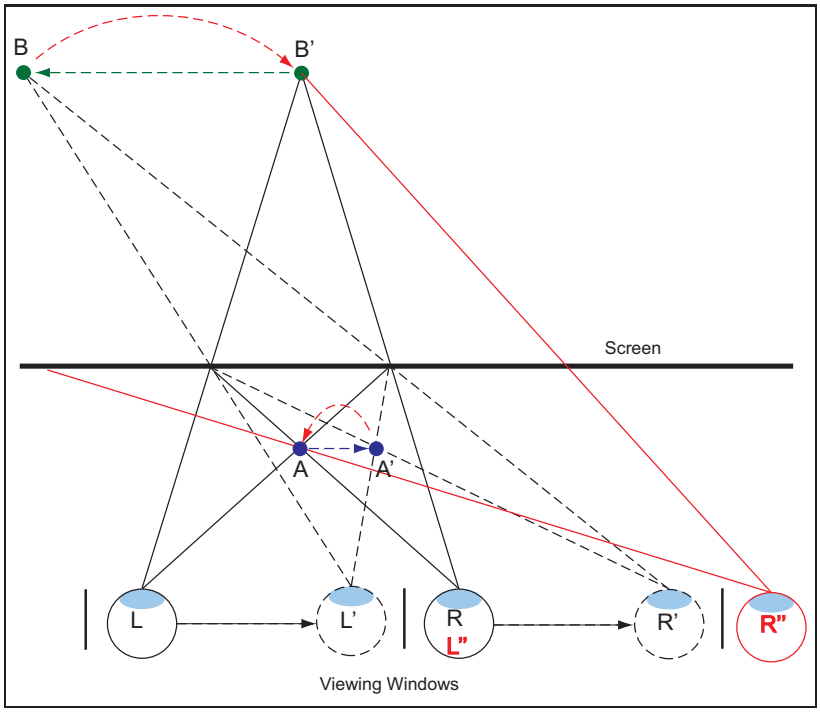
A basic parallax raster barrier is simply a screen composed of vertical opaque slits. The different perspective images are interlaced in columns and the barrier set-up in such a manner that each column can only be seen from a certain angle, thus forming distinct viewing regions. Figure 2.14 shows the arrangement of a raster barrier and LCD display for a 4-view 3D display. A significant advantage offered by the parallax barrier design is that they can easily be made by photolithography and printing techniques which can offer more accuracy than lenses [98].

For a two-view display the distance between each viewing window, E at the ideal viewing distance Z is usually equal to the average eye separation so as to give a small amount of viewing freedom. However, for multiview displays E is often set much smaller so as to avoid flipping and false rotation effects. In any case the barrier width, B_w can be determined for an N-view display from the following two equations using the principle of similar triangles:

$$\frac{B_w}{Z} = \frac{(N-1)P_w}{S+Z} \tag{2.3.1}$$

$$\frac{B_w}{Z} = \frac{(N-1)E}{S+Z} \tag{2.3.2}$$

Figure 2.13: As an observer's eyes move within a viewing window from position L and R to L' and R' perceived points in front of the display shear with the observer's lateral movement whereas points perceived behind the display shear in the opposite direction. As an observer moves into the next viewing window with their eyes at positions L" and R" they receive updated pixels and suddenly perceive the points back in the original position resulting in a flipping effect.



Rearranging equation 2.3.1 gives:

$$S = \frac{Z(N-1)P_w}{B_w} - Z \tag{2.3.3}$$

Substituting the result (equation 2.3.3) for S in equation 2.3.2 and rearranging gives:

$$B_w = \frac{N-1}{\frac{1}{E} + \frac{1}{P_w}} \tag{2.3.4}$$

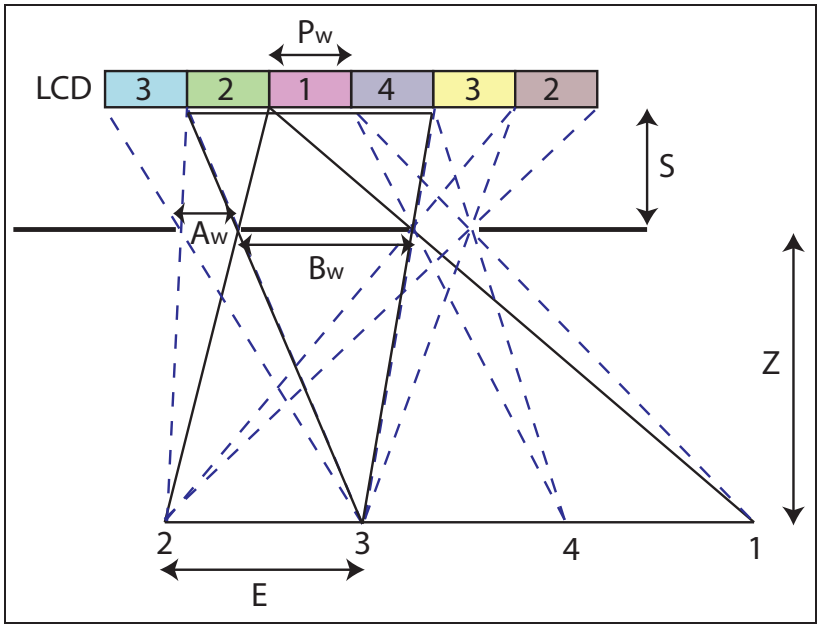
The viewing distance at which the width of the viewing windows will form at the desired E can be found with the following equation which is again derived by using similar triangles:

$$\frac{P_w}{S} = \frac{E}{Z} \tag{2.3.5}$$

which can be rearranged to give:

$$Z = \frac{E.S}{P_w} \tag{2.3.6}$$

Figure 2.14: A parallax raster barrier set-up to display four different views (see [168] for more details).



The barrier aperture, A_w can be derived in a similar fashion as equation 2.3.4 to give:

$$A_w = \frac{1}{\frac{1}{E} + \frac{1}{P_w}} \tag{2.3.7}$$

It can be seen in equation 2.3.6 that the optimum viewing distance is dictated by the width of each pixel, P_w and the thickness of the substrate, S between the barrier and LCD. Narrowing S can result in thin brittle glass sheets, which increases manufacturing and handling problems. Therefore, many auto-stereoscopic displays have viewing distances which are much further than typical desktop viewing distances, for example the X3D multiview display [127] has an ideal viewing distance of about 650 mm. The problem of viewing distance recedes as the number of views increases, because the window width tends to decrease as well.

Parallax barriers also block out a lot of light from the display and the glass surface can cause reflection [115]. However, these problems can be overcome by using bright light sources and anti-reflection coated optics [66].

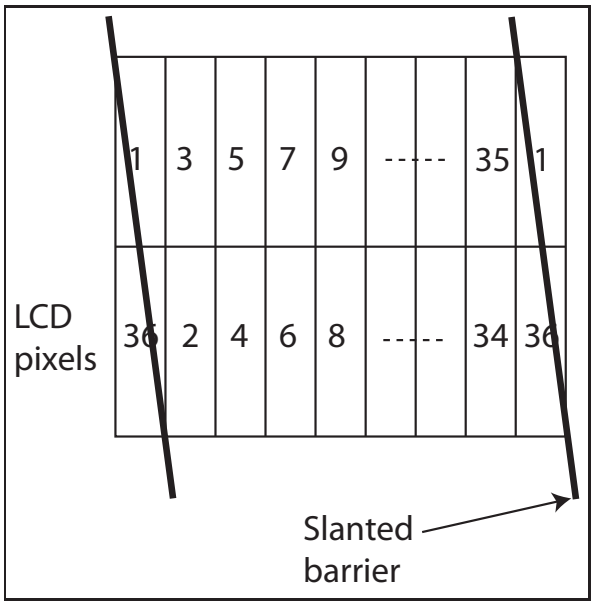
The resolution of each image is determined by the aperture pitch of the barrier, whereas the number of viewing windows is determined by the width of the opaque strip. Higher

quality images can be presented in each view by narrowing the width of the aperture and increasing the resolution of the LCD. However, as the width of the slits narrows diffraction becomes more apparent, which causes ghosting. The result is that either the number of viewing windows must be restrained to maintain high quality images or the resolution of each viewing window suffers detrimentally. This is considered the fundamental limitation of parallax barriers [67].

Philips 36-view display

An example of a multiview display using a parallax barrier is the Philips 15.1 inch 36-view display [98]. Philips tackle the problem of maintaining a high resolution per view by using an ultra-high resolution LCD of 3200x2400 pixels and slant the barrier at an angle of $\arctan(\frac{1}{6}) = 9.46$ degree, as shown in Figure 2.15, so as to share the pixels in the vertical and horizontal direction, thus keeping a good aspect ratio; the resolution per viewing window is 533x400 pixels.

Figure 2.15: The arrangement of the pixels for each viewing zone using a slanted parallax barrier design in the Philips 36-view multiview display [98].



With the slanted barrier arrangement, most pixels can be visible in two consecutive views. This has the benefit of disguising the black mask between the pixels which is

usually visible as a dark line at the boundaries of the viewing windows by blurring the views [66]. For a two-view display this amount of cross-talk would be generally considered very bad design as it can cause extreme visual discomfort and reduces the quality of the 3D image [92]. However, Lee et al. [98] argue in cases where the depth range is limited and parallax changes between each view small, cross-talk in multiview displays can actually increase the resolution per view and improves the 3D quality as it smooths the transition between each view, effectively decreasing the flipping effect. Philips engineers reported that the display produces high quality natural looking 3D images and they did not experience any visual discomfort.

A recent study [87] of cross-talk in multiview displays on perceived image quality showed that although cross-talk decreased subjective quality assessment, the cross-talk visibility threshold is higher than previous studies using two-view displays. However, the study varied the cross-talk by simulating the pixel structure of the philips lenticular multiview display using a 2D display. Therefore, the experiment did not correctly stimulate the disparity receptors as would a 3D display, and effectively blur based anti-aliasing in 2D was studied rather than 3D cross-talk.

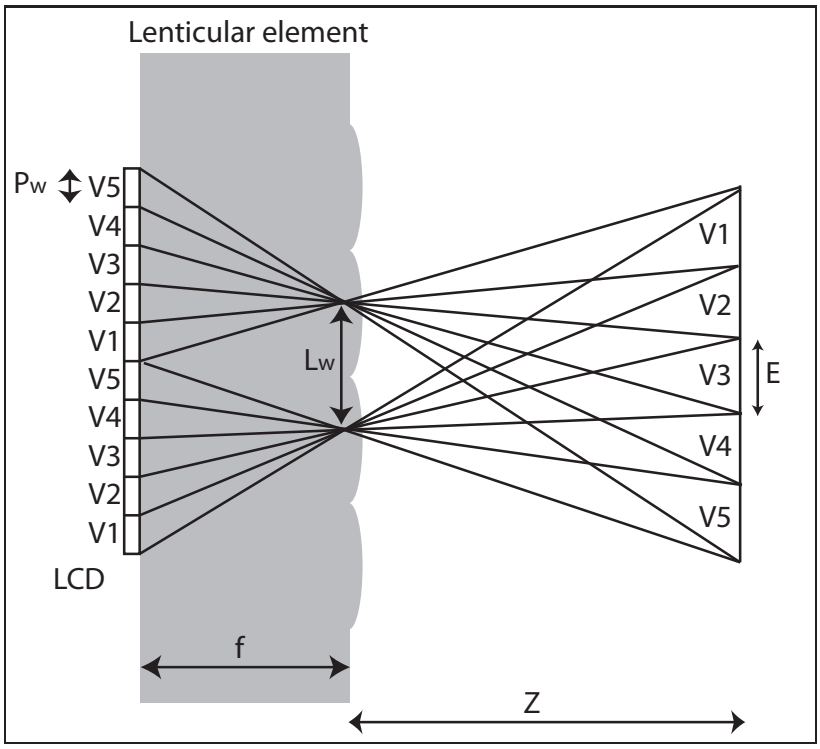
There is some evidence which suggest crosstalk is perceived as blur, similar to depth of focus in natural viewing, when the amounts of depth are very small [138]. However, a formal investigation is still required to quantify and validate both the quality of the Philips 36-view display (and other multiview displays) and the claim that cross-talk in multiview displays can improve the quality and resolution per view, and importantly the effects that this type of cross-talk has on comfortable depth limits.

Lenticular element design

First invented by Hess [63], vertical strips of tiny cylindrical lenses forming a sheet can be aligned on top of the display with the images appropriately interleaved to create multiple viewing windows and views. Figure 2.16 demonstrates a lenticular design for a 5-view display. They require more effort to build than parallax barriers but do not block as much light out from the display.

The optimum viewing distance can be found with the following equation which is

Figure 2.16: Lenticular lens array for a 5-view multiview display (diagram adapted from [66]).



derived by using similar triangles:

$$Z = \frac{E.f}{P_w} \tag{2.3.8}$$

It can be seen from equation 2.3.8 that the viewing distance is mostly determined by the focal length, f which is in turn dictated by the substrate thickness. Thus, lenticular elements suffer the same difficulty of controlling the minimum viewing distance as parallax barriers.

Other problems with lenticular technology are: the difficulty of applying an anti-reflection coating onto the lenses to avoid distracting reflections within the 3D image; the scattering of light on the irregular surface; and dirt becoming trapped between the lenses degrading overall image quality. The scattering of light is quite a serious problem as it makes the images appear misty [66] and the surface of the lenticular array can be distinguished from the underlying image by the naked eye. Also lenticular displays magnify the underlying device's subpixel structure and because of the black mask between the pixels, dark lines at the viewing window boundaries seem more apparent than with parallax barrier displays [36]. For these reasons it is has traditionally been quite difficult

to build a high quality lenticular 3D display.

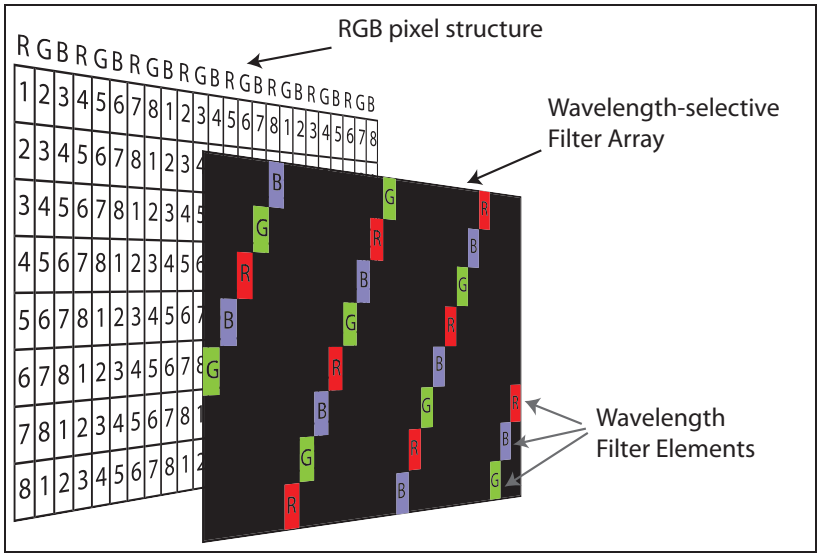
The problem of the visible boundaries can be alleviated by using a slanted lenticular design [156] similar to the Philips 36-view parallax barrier display [98]. This has the effect of blurring the black mask and pixels together to hide the structure. However, this increases cross-talk and imposes further limits on the comfortable depth range of the display. Sharp addressed this problem more successfully with a novel pixel configuration known as PIXCON [51]. The pixels are rectangular and horizontally contiguous so that the black mask used to normally cover the electronics between the pixels is completely removed.

Recent advances in micro-optic coating processes [61] have greatly improved the quality of the lenticular lens. The process involves filling in the irregular surface of the lenticular array with a low refractive index material such as plastic. A ”lens booster” is also incorporated into the substance so as to maintain the required high refractive index difference between the lens array and air. This effectively produces a flat lenticular array and almost eliminates the scattering effect.

Wavelength-selective filter array and Newsight Corporation

Newsight Corporation (formerly Opticality Corporation/X3D Technologies) have developed a number of multiview displays [127, 128] which are all based on a wavelength-selective filter array technology [133] designed by 4D-Vision. The filter is similar to a step parallax barrier [105] as it is composed of opaque strips arranged in a diagonal fashion, and thus shares many properties of the parallax barrier. The difference is that each rectangular aperture contains either a red, green or blue wavelength filter element. Figure 2.17 shows the arrangement of the sub-pixels and wavelength-selective filter array for the 8-view X3D display. The manufacturing costs are low and the filter is not sensitive to the calibration procedure, which allows the company to produce displays using LCD technology from only 2 to 50 inches and using projection technology to as large as 200 inches. The main disadvantage of the filter is its reduction of light, and therefore very bright light sources are required.

Figure 2.17: A wavelength-selective filter array set-up to produce 8 views (diagram adapted from [133]).

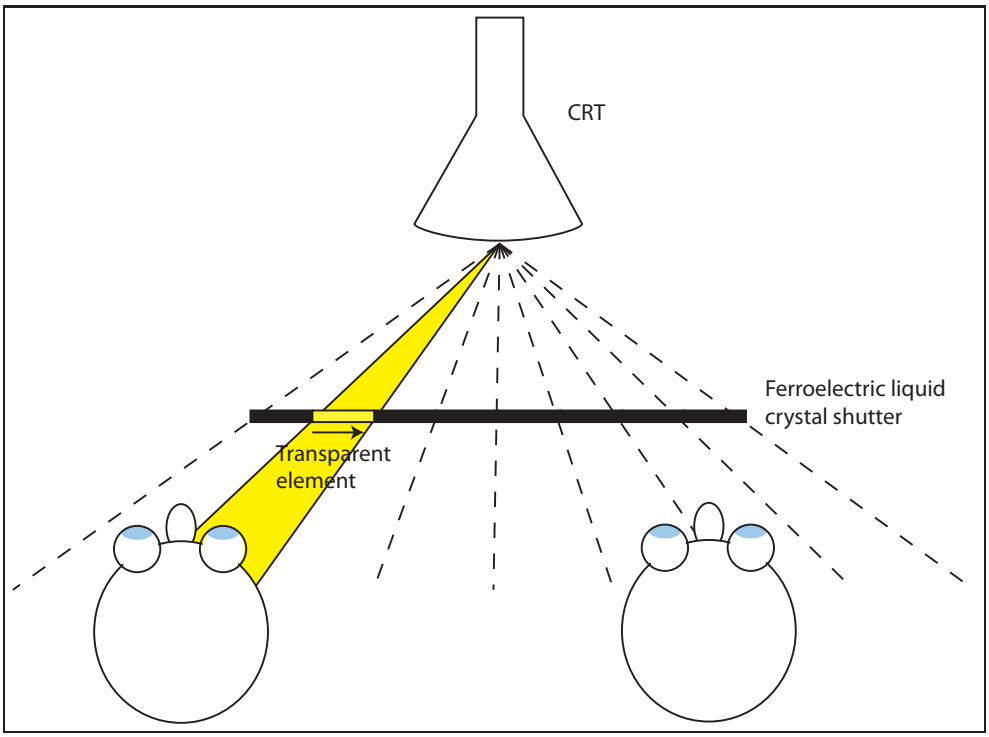


Cambridge display

The Cambridge multiview display [152,153] developed at the University of Cambridge is based on a time-sequential design using a very fast CRT display and ferroelectric liquid crystal shutters. Each view is produced by the CRT one at a time while the liquid crystal shutter effectively creates a moving slit, by turning the corresponding liquid crystal segment transparent, so as to direct that image into the appropriate viewing window as shown in Figure 2.18. A compound image transfer lens correctly focuses the image produced by the CRT onto the shutters, and a 10 inch diagonal Fresnel lens projects each image into space. The first Cambridge display was monochrome and capable of producing 16 views at a resolution of 320x240 or 8 views at 640x480 on a 10 inch diagonal screen. Colour was added in later Cambridge displays [112] by adding a Tektronix nematic liquid crystal colour shutter which dynamically filters the light from the monochrome CRT in a sequential fashion. This unfortunately has the effect of the dividing the maximum number of views available by three since each image has to be produced three times; once each in red, green and blue.

A ferroelectric shutter with a switching time of less than 100 micro-seconds is required to display the multiple views at a sufficiently high enough frequency to avoid flickering.

Figure 2.18: Principle of the moving slit in a Cambridge multiview display.



However, the ferroelectric shutter blocks light from the CRT by a factor of $\frac{T_p}{N}$, where T_p is the uncrossed transmission of the polarisers used by the ferroelectric shutter (approximately 0.35), and N is the number of different views [39]. As more views are added to the display, the brightness of each view decreases. For example, 96% of the light from the CRT is absorbed in the 8-view display and almost 98% absorbed in the 16-view display. This means very bright light sources are required.

The problem of brightness and low view resolution for the colour display was overcome in a more recent design by replacing the single monochrome CRT and colour filter with separate red, green, and blue CRTs [39]. The display is capable of producing 15 views on a 50 inch display at a resolution of 640x480 at 30 Hz with about 250 cd/m^2 luminance.

A problem unique to all time-sequential 3D displays is the possibility of temporal artifacts which are perceived as stereo aliasing effects [22]. This occurs when objects in the scene move horizontally and due to the sequential fashion the display updates each view, the left and right eyes will briefly view a stereo image which is out-of-synch. The

object in the out-of-synch stereo image will either display extra disparity or less. Multiple objects moving at different velocities can therefore lead to disturbing depth distortions. This problem can be ameliorated by increasing the frequency each view is display at; however, this must lead to a decrease in either the number of views or the resolution.

Mitsubishi multi-projection displays

Multiview projector systems use a single projector or display for each view which allows multiview displays to be built with a greater number of views and of higher resolution compared to other multiview technologies. Both lenticular and parallax barrier technology can be extended to incorporate multi-projectors. Mitsubishi have built front and rear 16 projector array displays using both techniques [9, 107]. The front projection system uses a single lenticular sheet and a retro-reflective front-projection screen material mounted onto the back. The main problem with the front-projection system is that the projector array takes up a lot of space and cannot easily be positioned such that the observer is able to be positioned in front of the display and not block the line-of-sight of the projectors. To avoid this problem a rear projection system was built using two lenticular sheets mounted back-to-back with an optical diffuser in the middle [107]. However, the double lenticular sheets must be aligned very precisely otherwise moiré effects will be observed. In practise this requires significant engineering effort.

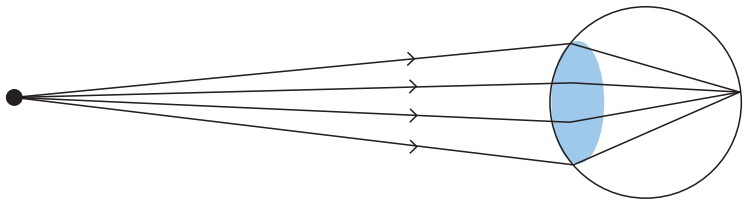
Synchronisation, alignment, colour and brightness differences between each projector is also an issue which requires sophisticated automation tools to solve. However, a greater problem is the sheer bandwidth required for some of these displays. Using 16 high-definition projectors produces 1920x1080x16 or more than 33 million pixels. This amount of data cannot be easily displayed, rendered/captured with cameras or transmitted without expensive specialised hardware.

Super-multiview

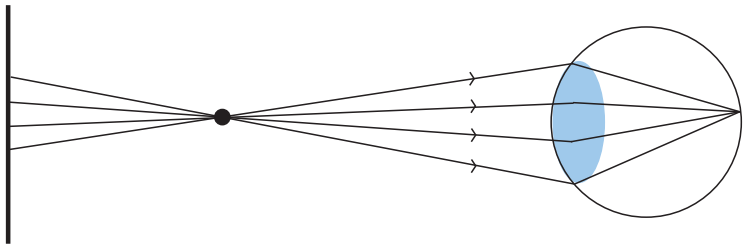
The different technologies available for creating auto-stereoscopic multiview displays can also be combined into hybrid systems that are capable of producing many more pixels and views. An interesting consequence of increasing the viewpoint density sufficiently so that at least two views enter the pupil is the potential to induce accommodation [67]; this

is because the pixels appear to be emitted as directional light beams rather than being scattered from the display in all directions (see Figure 2.19). These hybrid multiview displays have been given the term “super-multiview”.

Figure 2.19: A point in 3D space emits many directional beams which must be converged onto the retina by the lens to form a clear in focused image. In normal stereoscopic displays the light is scattered in many directions from the screen therefore, the eye must focus at the viewing distance regardless of any perceived depth. Super-multiview displays are capable of producing directional beams of light from each pixel so the correct accommodation will be induced for each perceived point in space.



(a) Accommodation of light projected from a point in space



(b) Super-multiview Display

The first prototype super-multiview display built was a red (monochrome) 45-view display with a resolution per view of 400 by 400 pixels and a refresh rate of 30 frames/sec [86]. In order to project the views into viewing windows narrower than the pupil diameter, the display utilised a Focused Light-source array (FLA) [85]. The basic concept of the FLA is to focus a number of different light sources, arranged in an arc, onto a single focal point by using beam shaping optics. The focal point of the FLA is then scanned rapidly by a mechanical X-Y scanner, while the intensity of each light source is modulated in correspondence to the correct pixel/position and perspective image.

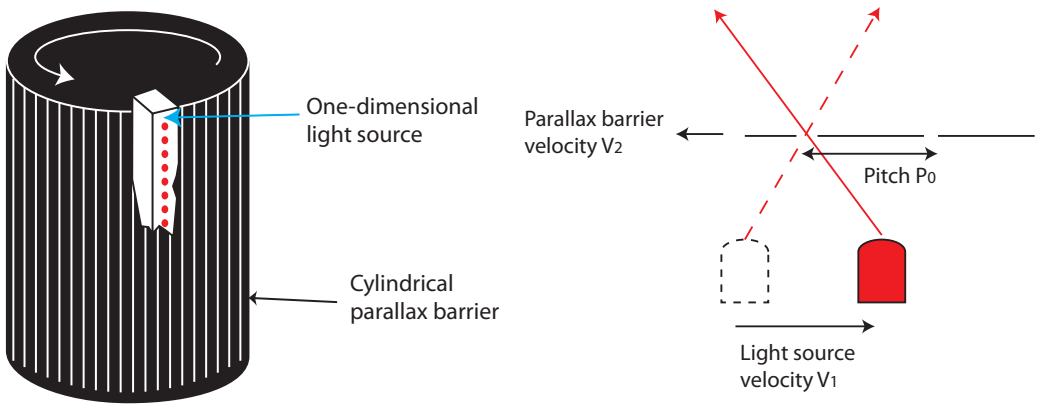
Another super-multiview display capable of 70 views, built by the Telecommunication Advancement Organization, is based on a parallax panoramagram using a cylindrical

parallax barrier [42] (see Figure 2.20(a)). The display is meant to replace the multiplex-hologram for public performances as it can provide a full 360-degree viewing angle with a refresh rate of 30Hz and thus capable of displaying moving images. The fundamental limitation of parallax barriers due to diffraction is overcome by rotating the cylindrical parallax barrier and a one-dimensional light source array in opposite directions, which effectively creates a virtual parallax barrier with a smaller pitch than the original barrier. Referring to Figure 2.20(b), the following equation more clearly expresses the relationship of the virtual pitch with actual pitch of the barrier:

$$P_v = \frac{V_1}{V_1 - V_2} P_0$$

Where P_v is the aperture pitch of the virtual parallax barrier, P_0 is the aperture pitch of the actual parallax barrier, $V_1 (> 0)$ is the velocity of the light source and the $V_2 (< 0)$ is the velocity of the parallax barrier.

Figure 2.20: The cylindrical multiview display [42] uses a parallax barrier rotating in an opposite direction to a one-dimensional light source which effectively creates a virtual parallax barrier with a smaller pitch than the original barrier.



(a) Cylindrical multiview display (b) Parallax barrier movement method

Displaying enough views to satisfy the super-multiview property is challenging due to the amount of bandwidth required. Therefore, existing designs of super-multiview displays do not simulate vertical disparity; however, this causes an accommodation mismatch between the apparent vertical and horizontal convergence point of the light rays.

Viewpoint density

How many views should a multiview display reproduce? Too few and problems such as the false rotation and flipping effect become apparent and there is a possibility of viewing comfort and the effectiveness of the motion parallax suffering detrimentally. However too many and each image suffers quality degradation due to the limited number of pixels available.

A number of investigations have been conducted to determine the optimum number of views required for multiview displays. Pastoor and Schenke [119] concluded that parallax discontinuities between views is the key performance factor and predicted (by extrapolating results) that for the full depth range utilized in 3D cinematography, 100 views per 10 cm are required for the quality to be rated good by novice observers. Pastoor and Schenke [119] also stated that observers will perceive the flipping effect more readily as depth and image contrast increases. However, the investigation does not systematically vary the contrast in each scene in a controlled fashion, and instead, the stimuli are composed of a selection of photographs with varying amounts of depth and one random-dot-stereogram. We feel the extrapolation of these results is unlikely to give the upper requirement on the number of views, for example given a scene containing more contrast than in any of the stimuli, more views would probably be needed. Also the experiment was carried out with a 3D display with a very low resolution of 256x128 pixels displaying black and white images at a viewing distance of 330 cm. These viewing conditions do not relate to modern 3D displays and therefore the results are probably of limited applicability today.

Speranza at al. [143] concentrated on determining viewpoint density based on observers' preferred perceived smoothness of viewpoint transition and recommended 80 views per 10 cm. The experiment made use of sparsely populated scenes with very simple shading (lack of high contrast textures) for the stimuli. However, as already stated by Pastoor and Schenke [119], parallax shifts are only readily perceived between points of high contrast. Also the use of a toed-in camera arrangement introduced false vertical disparity [108]. Vertical vergence is considered just as important as horizontal vergence to bring points into correspondence [114, 137]; therefore, incorrect vertical disparities in the stereo images will provide false depth cues and could potentially cause viewer dis-

comfort [108] and affect the subjects' ratings.

Runde [131] carried out an experiment which allowed each observer to adjust the 3D settings to subjectively achieve the most natural looking image reproduction and recommend at least 44 views per 10 cm for a maximum viewing distance of 80 cm (viewing distance varied in the experiment between 50-80 cm). Unfortunately the viewpoint density variable was confounded with a number of other variables such as the viewing distance and direction of head motion, and the scene depth was not specified; therefore, these results are difficult to interpret as generally applicable guidelines.

A common problem with these investigations on the issue of multiview viewpoint densities is that the conclusions were based on self reports, i.e. from observers answering questionnaires or giving scores from rating scales. Slater [141] argues self reports are unreliable since they cannot reflect the changing state of the participant during the ongoing experience and the questions themselves can affect the results. Subjective post-test measures are known to be unreliable since they allow inconsistencies across different raters and rating situations [77].

Summary and conclusion for multiview auto-stereoscopic displays

To summarise, most multiview displays are based on either spatial multiplexing, time sequential or multi-projector designs. The displays can be set-up to produce relatively few views such as only 8 with the goal of increasing viewing freedom. Alternatively the number of views can be maximised to improve the naturalness of the 3D viewing. Increasing the views offers a look-around effect with increasingly smooth parallax changes and ultimately induces correct accommodation. The pursuit for a more natural 3D experience significantly increases the cost of the system.

Analysis of the viewing windows for auto-stereoscopic displays reveals that two-view displays are not very practical and cannot work correctly for everyone [37]. Interpupillary distance (IPD) varies greatly amongst the population from approximately 40 to 80 mm [35]; this range covers everyone who could reasonably be expected to look at a display. Usually auto-stereoscopic display designers assume an average IPD of 65 mm. However, any deviation from this averaged value reduces the viewing freedom for the observer. An extreme example would be an observer with an IPD of 80 mm who would

only have a viewing freedom of 25 mm and can easily be positioned so that a monoscopic image is perceived.

Dodgson [37] argues that a three-view display with head-tracking and a viewing window width of 40 mm can however work correctly for everyone with an IPD within the range of 40 to 80 mm. Examples of three-view head-tracking auto-stereoscopic displays can be found in [169]. Dodgson also argues that multiview displays which overlap the viewing zones to ameliorate the visible viewing boundaries must produce about six views and incorporate head-tracking to work for everyone. It can be concluded that if viewing freedom is important, then multiview displays with or without head-tracking are the most promising solution.

Some of today's most commercially successful multiview displays are lenticular or parallax barrier based using either LCD or plasma screens [158]. DisplaySearch, a display market research company, determined that flat panel display price per square-metre for TV panels has declined 25% per year since 2003 [34]. Coupled with such drastic price cuts is a steady improvement in the quality and resolution. Therefore, it is expected that multiview displays will become increasingly more attractive and affordable to the consumer.

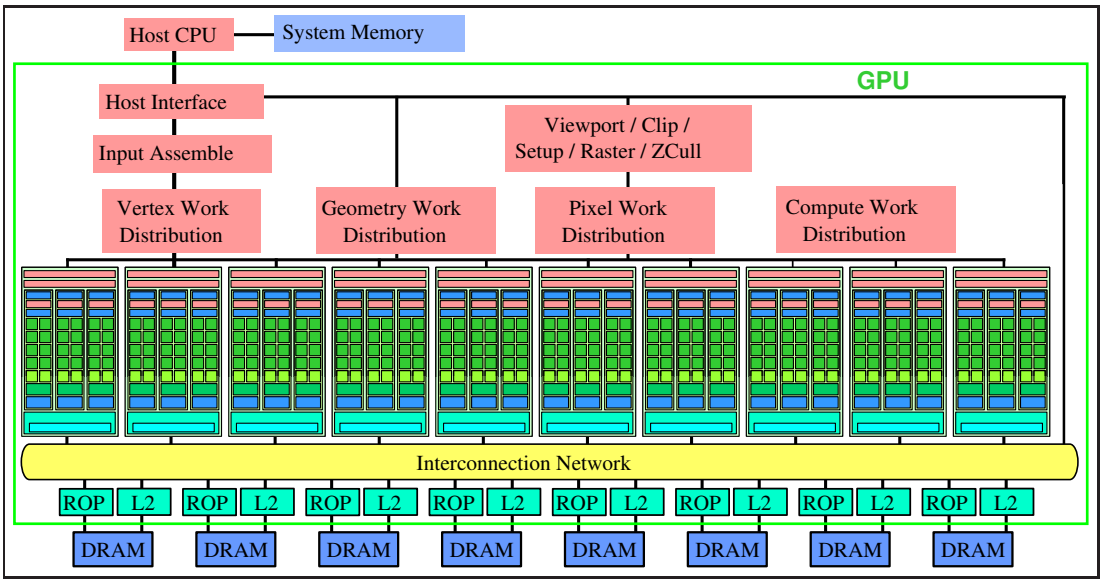
2.4 Rendering three-dimensional content for stereo displays

Since the 1980s field of computer graphics has rapidly grown in sophistication and interest largely due to the ever increasing computational power coupled with decreases in price. This section gives an overview of the steps required to convert a computer model into a 2D image as well as a stereoscopic image. A computer model is usually three-dimensional and contains description of the geometry, materials, lighting, viewpoint, actors, etc.

2.4.1 Rendering pipeline (OpenGL)

Today, most of the rendering is offloaded to the graphics card or graphics processing unit. The GPU is a very specialised piece of hardware designed specifically for graphics ren-

Figure 2.21: A simplified diagram of the NVIDIA GT200 GPU architecture.



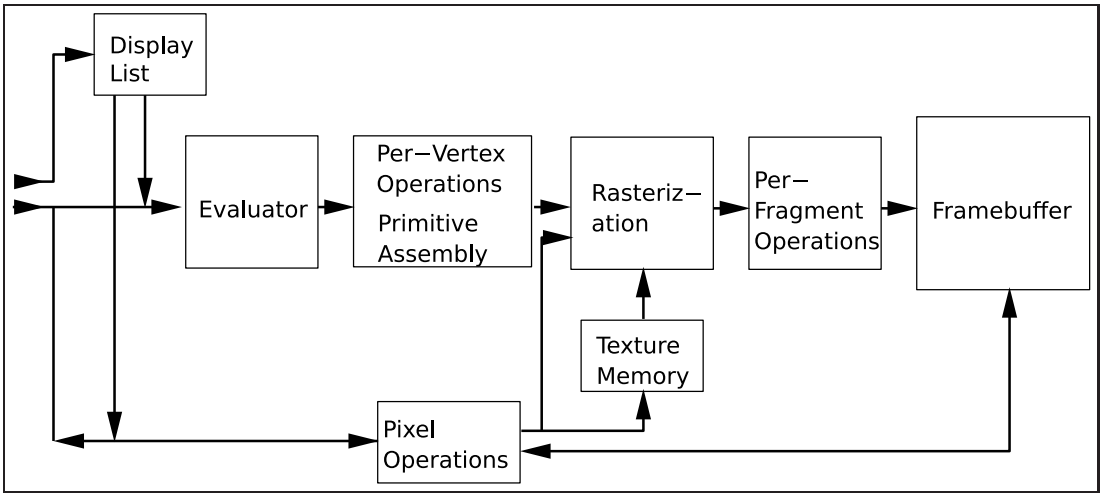
dering and is often much more powerful, with respect to the number of mathematical operations it can perform, than the central processing unit (CPU) of the computer it resides in. The GPU can be considered a single instruction multiple data (SIMD) stream processor, which processes a stream of vertices through a number of different stages or a pipeline resulting in a rasterised image stored in framebuffer memory ready to be displayed. The hardware design of GPU's can be incredibly complex: Figure 2.21 illustrates a simplified diagram of NVIDIA's GT200 compute architecture. Usually communication with the graphics hardware is performed through a standardised graphics APIs, of which the two most popular are OpenGL [4] and DirectX [1]. Since both APIs interface with the same hardware, they can generally be considered equivalent in functionality.

Figure 2.22 illustrates a simplified overview of how OpenGL processes data. Commands are sent from the left and proceed through the processing pipeline. The commands can specify geometric objects to be drawn or control how the objects are handled at various stages. The display list can be used to accumulate commands to be processed at later time.

The first stage is responsible for approximating smooth curved surfaces and geometry by evaluating certain mathematical functions. The second stage deals with geometric primitives described by vertices which are grouped into either points, line segments or

polygons. In this stage vertices are transformed with rotations and scaling functions, projected, and finally the primitives are clipped to the viewing volume. In a fixed OpenGL rendering pipeline, lighting calculations would also be performed on each vertex. The rasterization stage converts each primitive into a series of 2D framebuffer coordinates, also known as pixel fragments. Each fragment is sent to the next stage for a number of per-fragment operations which include z-buffering or depth culling, blending with other stored colours, and stencil testing, which allows testing of the fragment's value against the stencil buffer for conditional updates. Data can also be sent to the pipeline already in the form of fragments, e.g. textures, skipping a number of stages. For more details on OpenGL refer to [14] whereas for a more general introduction to computer graphics see [47].

Figure 2.22: Overview of OpenGL's rendering pipeline (from OpenGL specification). Commands enter from the left and proceed through the pipeline.



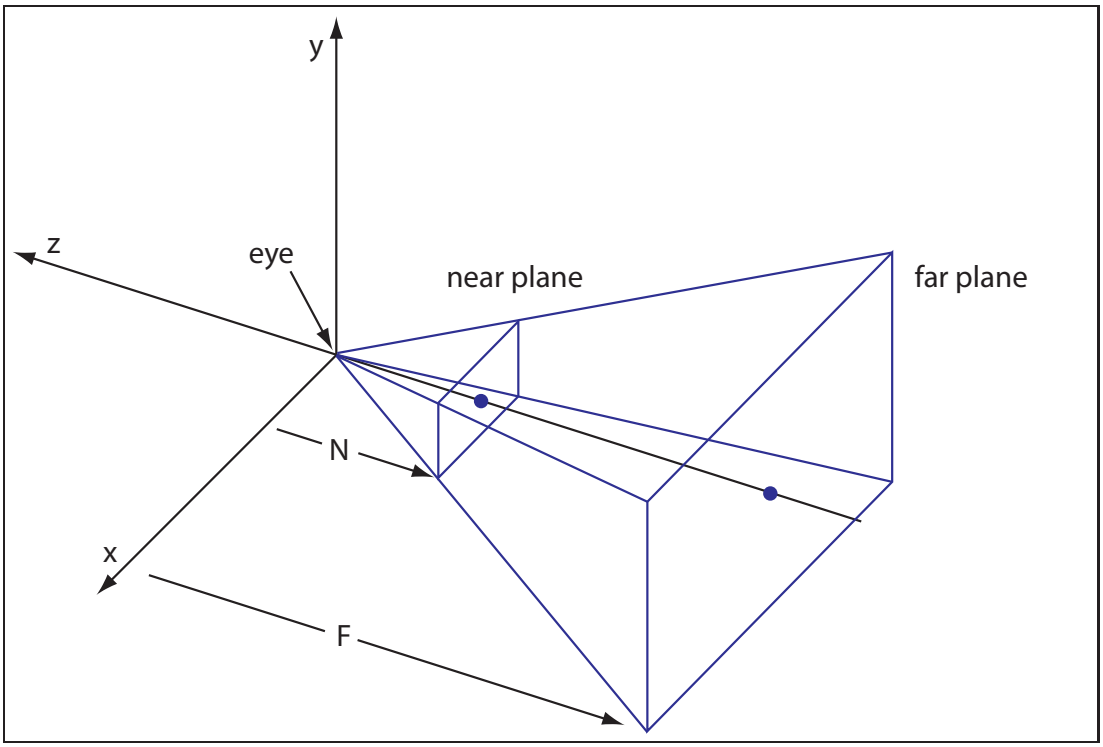
Programmable pipeline

Over the past few years commodity GPUs have evolved from only implementing a fixed-function rendering pipeline to an increasingly flexible programmable pipeline. User developed programs known, as shaders, can replace sections of the vertex and fragment operations stages. Programming the GPU can be achieved using a number of shading languages, for example: ARB low-level assembly language which is designed to be used

with OpenGL; OpenGL shading language [5] (GLSL), which abstracts from the underlying assembly language or hardware-specific languages and is based on the C programming language; Cg programming language [3] developed by NVIDIA which is API independent; and DirectX High-Level Shader Language [2] (HLSL) which is similar to GLSL but is designed to be used with DirectX.

The main advantage of using programmable shaders is that the functionality of graphics APIs can be greatly increased, often in ways unexpected by the GPU manufactures; for example, a relatively modern field of research called general-purpose computing on graphics processing units (GPGPU) attempts to harness the parallel processing power of GPUs to perform computation traditionally handled by the CPU.

Figure 2.23: The canonical camera is a virtual camera which describes the viewing volume within the computer 3D model by using cartesian coordinates.

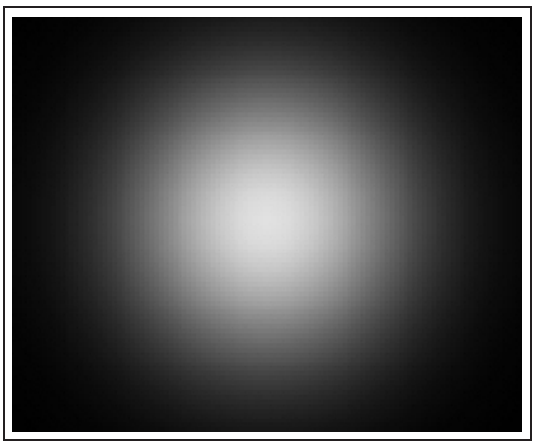


2.4.2 Rendering primitives

Triangles

While OpenGL allows the drawing of a wide range of primitives–for example, quadrilaterals, n-gons, lines, points, etc–triangles are by far the most widely used primitive in computer graphics. The reason is simple: most 3D objects and shapes can be decomposed into a number of triangles; therefore, it is more efficient to minimise the number of primitive types to process by decomposing as many objects in the scene as possible to triangles. Furthermore, the rendering pipeline is optimised for triangle processing [47].

Figure 2.24: A Gaussian splat.



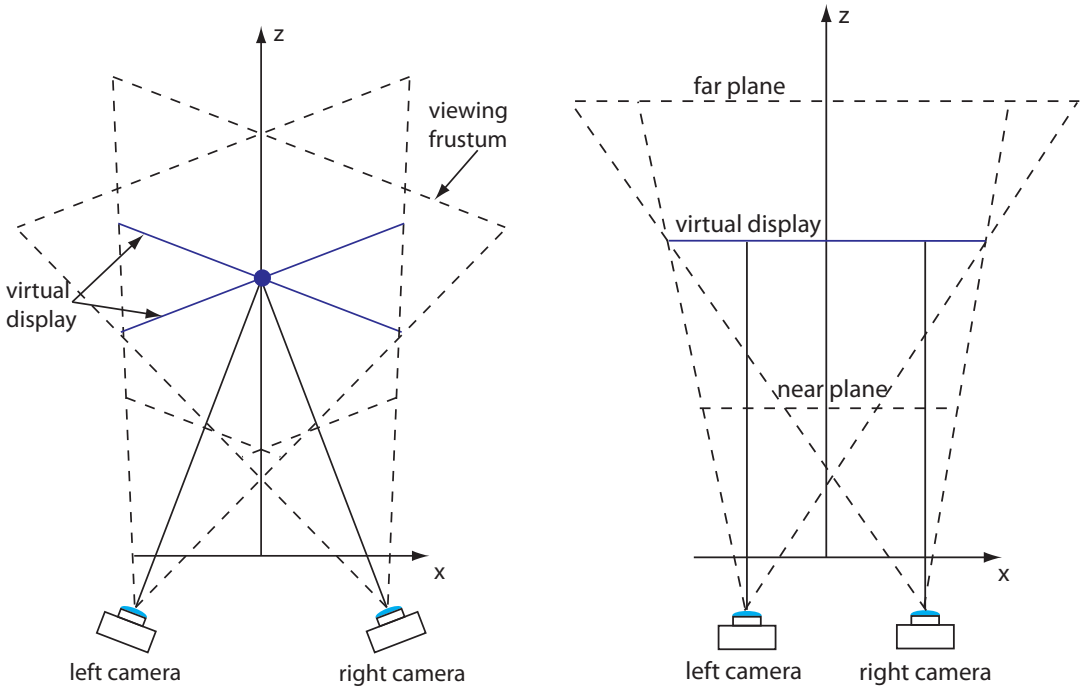
Point-based rendering

In the case of highly detailed models (when the projected primitives are smaller than the pixels of the display screen), cloud-like structures, or particle data sets, polygonal primitives such as triangles triangles become inefficient and point-based rendering (PBR) [99, 132, 165] techniques are more appropriate. The classical PBR method for representing points is by using viewer-oriented billboards or sprites which are texture mapped with alpha-blended Gaussian cloud-like textures [159]; these textures can also be referred to as splats (see Figure 2.24 for an example).

2.4.3 Camera models (monoscopic and stereoscopic)

In the real world, capturing a picture requires a camera; in computer graphics, a virtual pinhole camera can also be set-up to render a monoscopic image from a 3D computer model. The virtual camera is described by a canonical viewing frustum in cartesian co-ordinates as shown in Figure 2.23. Projection of the vertices within the volume onto the image plane requires a perspective projection matrix transformation.

Figure 2.25: Stereo pair virtual camera set-up for (a) toed-in and (b) parallel configuration.



(a) Toed-in camera model

(b) Parallel camera model

The simplest method of rendering a stereoscopic image pair is to set-up two virtual cameras with either toed-in view vectors or a parallel axial offset of the cameras. Both camera models are shown in Figure 2.25 and the geometrical parameters and possible distortions are described and analysed in detail in [170]. In the toed-in camera model, both cameras converge towards a point of interest; this point will be displayed with zero parallax and appear at the display screen. A parallel camera model requires either horizontal shifts of the CCD sensors in the camera or shifts of the images displayed on the monitor. Since the camera is actually virtual the same effect as a CCD sensor shift can be achieved

by using two asymmetrical frustums which coincide with the virtual display.

The toed-in camera model introduces curvature of the depth plane and keystone distortion which can lead to objects in the corners appearing further away than objects closer to the centre of the display (for objects lying on the same depth plane). Keystone distortion also causes vertical parallax which, even in small amounts, can cause considerable visual discomfort and limit the fusible depth ranges [170]. The parallel camera model does not generate keystone distortion or vertical parallax and is widely recommended over the toed-in camera set-up [83].

2.4.4 Controlling the amount of perceived depth

Producing a comfortable stereoscopic image manually involves a lot of trial and error is often tedious due to the large number of parameters that can be tweaked: camera separation, virtual display distance, field of view, and distances to the near and far clip planes. Fortunately, a number of methods have been devised which can automatically calculate the stereoscopic parameters required for the desired amounts of perceived depth; these techniques are discussed below.

Wartell [163] describes a complicated method of pre-distorting the scene before applying an equation that calculates the camera separation using a false eye separation. The false eye separation is set to an amount much smaller than the average human interocular distance so as to reduce disparity and avoid discomfort. The furthest depth plane is then described by using the false eye separation as the maximum screen disparity allowed, and scene depth is compressed within this volume. While Wartell's method eliminates the shearing distortions in a head-tracked system, depth compression of the scene varies as the observer moves in a direction perpendicular to the display. Also, this method does not allow the user to control the perceived depth in a precise manner since the observer's real eye separation is not taken into consideration and using a false eye separation can not be used to intuitively map the scene to a desired volume of perceived depth.

Jones et al. [83] describe a much simpler method of controlling the perceived depth by clearly distinguishing between viewer and virtual space and then calculating a transformation between the two spaces in a manner that allows precise control of the mapping of scene depth to perceived display depth. Furthermore, depth distortions such as shear-

ing and keystone are removed, and although depth compression is inevitable, at least it is constant, regardless of observer movements.

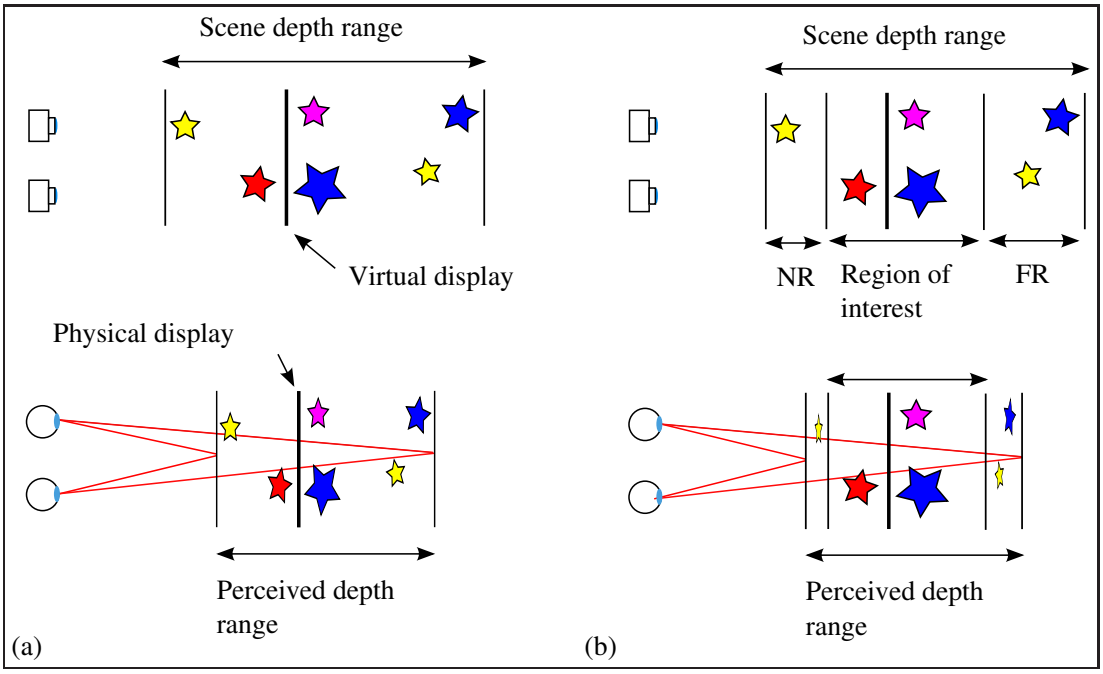
Ware et al. [161] proposed a two-stage process of scaling the scene and then adjusting the camera separation dynamically based on average camera separation preferences from their human factor investigation. Although depth distortions vary as the camera separation changes dynamically, it was argued that observers did not notice this if the rate of change in camera separation was less than 0.2 cm/sec. However, a fundamental flaw with the algorithm is that the camera separation is dependent on preferences of the subjects in the original experiment; the potential for a large range of eye separations, viewing conditions and display sizes was not taken into consideration.

William and Parish [167] developed a piece-wise linear algorithm which can map the scene volume arbitrarily to the available disparity so that regions of interest can be presented with more depth; Figure 2.26 illustrates this concept. The algorithm takes into account screen dimensions, desired perceived depth and the observer's eye separation. However, head-tracking is not taken into consideration and so various distortions will occur during head movements with or without head-tracking perspective updates. Unfortunately, implementation details have not been described very well and there are no results which can be used for evaluation purposes.

Holliman [64] also developed a piece-wise linear algorithm, but extended it from the work of Jones et al. [83] so that depth compression is constant and all other distortions are removed. Further work to smooth possible visual discontinuities of objects crossing different disparity region boundaries was also conducted [65]. Although additional computation is required, shading artifacts at the region boundaries are noticeably reduced and the disparity gradient at the regions is much smoother, thus allowing more depth to be fused by the observer.

To date, no depth control algorithms specifically designed for multiview displays have been reported. However, two-view depth control algorithms such as [64, 83] can easily be extended to the multiview case (further explanation is provided in Section 4.3.2).

Figure 2.26: The perceived stereoscopic depth on a 3D display is often different from the scene depth. (a) Camera models can be used to compress or map the scene depth to the confortable depth range of the display. (b) Holliman [64] describes a method of sperating the scene into three different regions, near region (NR), region of interest (ROI), and far region (FR) which can be mapped independentaly to the available stereoscopic depth (diagram adapted from [64]).



2.4.5 Aliasing problems and anti-aliasing methods

A problem common in computer graphics is a phenomena known as aliasing. This section briefly describes the problem as well as some common anti-aliasing techniques (see [47, chap. 14] for more details).

Examples of well known aliasing artifices are jagged edges (or 'jaggies'), moire fringe patterns (see Figure 2.27(a)), lost detail, disappearance of small objects, breaking up of long thin objects, flickering, etc. Anti-aliasing techniques can be applied to ameliorate these aliasing effects (see Figure 2.27(b) for an example). However, in order to understand the causes of aliasing, it is useful to know the basic concepts of signal processing.

Rendered or photographically captured images can be represented by a 2D signal or function, for example, Figure 2.28 illustrates a 1D signal representing the intensity variation along a scan-line of an image. Signals can be defined as continuous or discrete.

Figure 2.27: Aliasing can often be seen in the form of moire effects as shown here in image (a) of a parrot's wing. Anti-aliasing methods such as applying a gaussian filter and then downsampling can ameliorate aliasing artifices, image (b). (image (a) is taken from [46] and image (b) was processed using Adobe Photoshop)

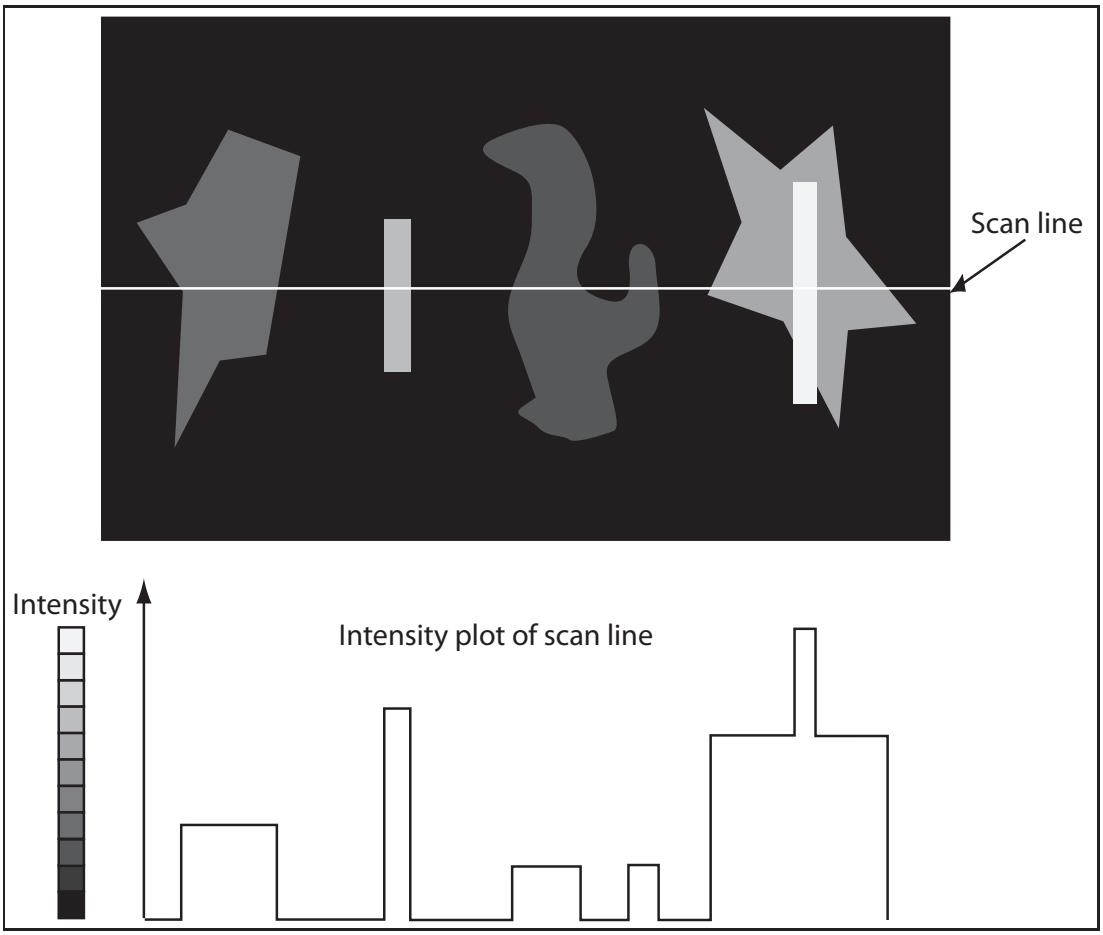
(a) Moire effect

(b) Anti-aliased image

With a continuous signal, the signal value can be found at each infinitesimal point within the domain of the function. A discrete signal on the other hand, is a sequence of val-

ues. A continuous signal can be converted to a discrete signal by sampling, whereas the reverse operation, known as reconstruction, is achieved by interpolating between the samples. The rendering pipeline ultimately produces a discretely sampled 2D signal (an array of pixels) and the display hardware is responsible for attempting to reconstruct the original continuous signal. The sampling approach taken determines how faithful the reconstructed signal will be to the original continuous signal.

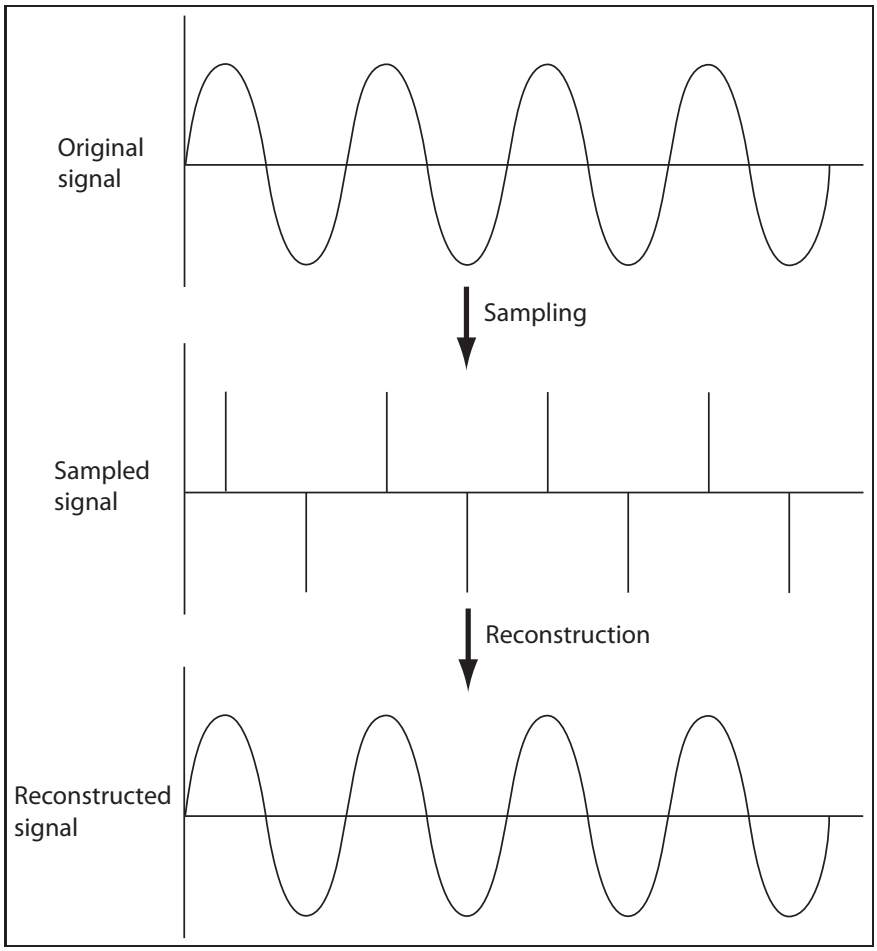
Figure 2.28: A scan-line of an image can be thought of as a 1D signal. The scan line is presented as an intensity plot.



Generally there are two methods of sampling: point sampling and area sampling. In point sampling, the value of each pixel is determined by treating the pixel as a point and evaluating the original signal at that point. The main problem with point sampling is that the points may not cover all the objects, especially if the objects are small. Increasing the sampling rate is a simple method of improving the reconstruction stage. Super-

sampling averages the values evaluated from a number of sub-pixels for each larger pixel and achieves better results but at the expense of more computation.

Figure 2.29: The process or pipeline of reconstructing a continuous signal from the sampled signal.



Area sampling eliminates the problem of missing objects in the sampling stage by integrating the signal over a square and averaging it for each pixel. A problem with unweighted area sampling is that if an object wholly contained within a pixel moves about while remaining inside the pixel, the intensity is constant; however, as soon as the object crosses into the next pixel, both pixel intensities are suddenly affected. Overlapping weighted functions can be applied to area sampling to correct this though. Unfortunately area sampling is not always practical since we cannot always integrate the signal, for example in ray tracing each pixel must be point sampled instead.

Figure 2.29 shows a continuous signal being point sampled and then reconstructed perfectly. In order for this to occur, according to sampling theory, the original signal must be sampled at a frequency of at least twice f_h: the highest frequency component present in the signal, which is also known as the Nyquist frequency. Sampling below the Nyquist frequency can lead to reconstructed signals of lower frequencies than the original and is the reason for aliasing artifacts. Signals with discontinuities such as sharp edges or object boundaries in the scene have an infinite frequency spectrum; therefore, while increasing the sampling rate can ameliorate aliasing, perfect signal reconstruction can never be guaranteed. Bandwidth limiting, which is also known as low-pass filtering, can be employed to remove high frequency components so that the signal can be reconstructed correctly from a finite number of samples. However, too much bandwidth limiting tends to blur the image as sharp details can often only be captured by high frequencies.

In stereoscopic images, a number of unique aliasing artifacts can occur which do not manifest in monoscopic images. Pfautz [125] identifies the following aliasing artifacts in stereoscopic perspective images due to spatial sampling:

- Inaccuracy in projected position.
- Inaccuracy in projected size.
- Inaccuracy in disparity.
- Inconsistency in projected size.
- Inconsistency in disparity.
- Inconsistency in disparity of horizontal edges.
- Inconsistency in position.

Projection of a single point in an image can result in up to half a pixel error in either vertical or horizontal directions due to the location being rounded to the nearest pixel. In 3D, disparity inaccuracies of up to a pixel can occur, resulting in inaccurate depth perception, and vertical parallax of up to a half a pixel will be present. As a point recedes into the distance, the pixel rate of movement towards the vanishing point is a function of its distance from the line of sight; however, this means the position of a line's two end points

may vary leading to inaccuracies and inconsistencies in the shape of an object. These aliasing artifacts should be taken into consideration when designing stereo algorithms if good quality images are to be produced.

2.5 Multiview rendering algorithms

Generating *n* views for a multiview display does not necessarily require n times the work of rendering a single view. Algorithms have been developed to exploit similarities in the scene between different perspective views–for example, stereo-perspective coherence, or temporal coherence–which can improve rendering performance. These algorithms can be broadly categorised into three groups: 2D or photographic image processing algorithms; 3D model rendering algorithms; and general coherence algorithms.

Image processing algorithms can be applied to generate new viewpoints from a set of previously rendered or captured images via interpolation techniques, e.g. [25, 55, 56, 178]. Fehn [44] has shown that new viewpoints can be generated from a single view if there is an accompanying per-pixel depth map; this method is known as depth-image-based-rendering (DIBR). The pixels are reprojected into the 3D space according to their respective depths and then projected into the image plane for each virtual camera and is known as image warping. A problem with point sampling and then reprojecting the pixels is that the resulting image may be different from correctly point sampling reprojected geometry. This is because point sampling only approximates the geometry and any errors will propagate if the pixels are reprojected. Vázquez [157] proposed a forward-mapping mechanism, where the newly rendered images are sampled on an irregular sampling grid and processed with disparity-compensated interpolation techniques to get a regularly sampled image, which minimises geometrical distortions due to sampling.

A potential advantage of depth-image-based-rendering is that existing 2D digital TV framework could be used to broadcast 3DTV. Also content can be easily obtained from 2D-to-3D conversion techniques [88], which obtain structure from motion. Another nice feature is the ability to increase or decrease the perceived depth according to the user's preference. However, obtaining the desired results may still be difficult since the stereo parameters have to be adjusted manually. Unfortunately, even if sampling errors could

be completely corrected, DIBR still suffers from not being able to handle occlusions and transparency properly. Parts of the scene which are occluded in the initial viewpoint, can become visible from a different viewpoint, but there is no data for these newly revealed parts of the scene resulting in holes or gaps in the rendered image. The holes can be filled in with interpolation techniques, but this tends to look poor. Also, since each pixel only has one assigned depth to it in the depth map, there is no way of implementing transparency.

Generating new viewpoints with image processing techniques can be advantageous because rendering times are usually not dependent on the scene complexity and therefore they scale better with larger data sets. Unfortunately image interpolation usually results in views which contain overlapping pixels or gaps and incorrect view dependent scene changes, for example specular highlights. One solution to the occlusion and transparency problem for depth-image-based-rendering, is to sample every surface in the line of site from the one viewpoint and store these samples as a 2D array of layered depth pixels [136]. However, the algorithm then becomes dependent on the scene complexity, and can become much less efficient than depth-image-based-rendering with only one depth map. Also the algorithm will still suffer from the sampling errors associated with reprojection.

Three-dimensional model rendering algorithms such as [24, 76, 118] attempt to exploit, or share the calculations across all the views used in rendering the images from a 3D model or data set. For example, scan-line rasterisation can be extended to volume render each particle for stereoscopic viewing in an order which maximises the sharing of projection calculations between all the points and the two views [118]. Castle [24] takes this one step further in the polygon scan conversion stage of rendering by incrementing the projection of each point by a fixed amount (disparity calculated for that particular depth) for each consecutive view and interpolating the intersection of the polygon edges with each scan-line.

Ray-tracing algorithms can also take advantage of calculation sharing to improve performance. For example, one method is to volume render the left view in a conventional manner, sampled points along each ray are then reprojected using a viewing transformation matrix [6]. Since early ray termination may result in samples which should be visible

in the right view but are not reprojected, some parts of the rays in the right view still need to be traversed. Time savings of up to 80% have been reported with this method. However, the right view will not be rendered entirely correctly due to approximations used in the reprojection calculations.

One algorithm which is technically a hybrid of computer graphics and image processing is [60]. The algorithm decomposes the geometric scene into special primitives which are used to render slices or epipolar plane images so as to form the spatio-perspective volume. An important property of epipolar plane images is that polygons form linear tracks over the different viewpoints; therefore, linear interpolation techniques can be used to render them without the usual accuracy problems associated with image interpolation techniques. Also the tracks in the epipolar plane images are often long, so the pixel to vertex ratio is high. This means less vertices are used to describe the geometry for many views than when traditional triangles are used; therefore, fewer calculations (e.g. lighting and projection) are required to obtain the final pixel colours. Results using this algorithm show that performance gains of one to two orders of magnitude over conventional single view rendering can be achieved. However, set-up costs of the spatio-perspective volume can mean little or no advantage gains for rendering a small number of views.

In very large data sets or scenes of high complexity, the size of the polygons can be smaller than the pixels which represent them. The ratio of the pixel size to the number of vertices is so small that polygonal rendering methods are not suitable; other methods such as volume rendering, ray-tracing and point-based-rendering (PBR) can be more effective. Therefore, for large data sets it is likely that the algorithm described by [60] will not be as effective as it was for triangle-based scenes.

Hübner et al. describe a texture splatting algorithm designed specifically for point based rendering on multiview displays. The algorithm takes advantage of the programmability of the GPU and for each point generates an enlarged quad texture to store all the splat projections from the different viewpoints before blending them all correctly for each view. The algorithm's efficiency is based on the fact that the geometry is only sent to the GPU and processed once rather than n times for a n-view display. However, the performance of the algorithm is unpredictable because the size of the enlarged texture quad and therefore the number of pixel fragments it contains varies depending on the point's'

distance to the focal plane and the resulting maximum disparity between the left-most and right-most view.

Another attempt at reducing the number of vertices which need processing is described by [32]. The scene is initially set-up for the left view and then the primitives, e.g. triangles, points, etc, are duplicated and transformed to generate the right view. The replication stage only occurs after the vertex processing stage in the programmable pipeline, therefore saving on some computation. Due to certain implementation constraints, z-buffering is disabled; therefore either depth sorting and painter's algorithm is employed or an additional depth map for the left view is rendered and used to approximate the visibility of the fragments in the right view. Both methods suffer from visual artifacts and are inefficient compared with standard z-buffering. The algorithm can only save on computation at the vertex stage and is therefore only beneficial if there is a large proportion of work, e.g. expensive lighting effects, at the vertex stage.

Rendering multiple views directly from a 3D model is usually more accurate than image processing algorithms and can also take into account view dependent lighting effects; therefore, they are more suitable for scientific applications, where accuracy can be of critical importance, than 2D processing techniques.

General coherence algorithms attempt to take advantage of temporal coherence as well as spatial coherence between the different frames. For example both the Talisman graphics architecture [151] and [104] have shown performance gains when rendering certain scenes. Structure from motion can be retrieved from various frames and used to synthesise novel views if the velocity of the camera is known. This method is demonstrated in [178], which overlays a regular triangle mesh over one view and warps the triangles with an affine transformation in relation to their associated disparity. Geometric distortions can arise though, when triangles contain more than one object at different depths. This problem can be reduced with edge detection so that the triangles can be matched to only one object [56]. Since the object transformations and camera velocities within a scene can vary greatly, the coherence between each frame and data set also varies greatly. Therefore, performance of general coherence algorithms cannot be predicted reliably and are difficult to use effectively.

As well as optimising algorithms to share computation across the views, significant

work has also been carried out on developing parallel rendering architectures for both new hardware [62, 147] and current off the shelf parallel systems [8, 94, 172], e.g. cluster PC's. These solutions are expensive though and are typically only used in cases where hundreds or even thousands of viewpoints need to be rendered.

2.6 Summary

In this chapter the relevant background in three broad topics were presented, which were: the human visual system and depth perception; three-dimensional display technologies; and the field of computer graphics. To summarise briefly, stereoscopic 3D displays aim to emulate a more natural viewing experience by presenting a different perspective image to each eye. The horizontal binocular disparity reproduced by these displays is a powerful depth cue that allows relative depth judgments to be made. Multiview 3D displays are auto-stereoscopic displays that produce more than two views simultaneously. These additional views provide a wider viewing angle for the display, allow a look-around effect and can easily support multiple observers. As a result multi-view displays naturally support both the stereoscopic depth cue and head-based motion parallax and could be superior for certain tasks.

The key findings were:

- The multiview approach has significant benefits but is also costly, it requires a display design that spatially or temporally separates the views; therefore, a particular issue for content creators and distributors is how to provide many simultaneous views of a scene.

- Point data sets are rapidly growing in popularity and complexity. Stereopsis can aid scientific analysis of these rich and complex data sets; however, little research has been directed towards efficiently rendering point data sets for stereoscopic viewing.

- In order to minimise visual discomfort and fatigue, simulated depth ranges should be restricted to within ± 10 cm for standard desktop viewing conditions; these limits can be relaxed as the size of the display and viewing distance grows.

- GPD is the most appropriate tool for measuring the amount of stereoscopic depth because it successfully accounts for viewing distance; GPD is the sole measure used in the remaining chapters of this thesis.

Chapter 3

Investigating performance of path searching tasks in depth on multiview displays

The background chapter discussed a number of different multiview displays, including commercially available displays which range from the order of ten simultaneous views to research prototypes capable of over a hundred views. Previous work [119] suggests that the number of views may need to be as high as 100 views/10 cm at the eye with the result that at any one time the display will be generating many visually redundant views. This is a significant optical and computational challenge for any system. We identified a number of other studies in the background chapter, all of which generally recommend a high number of views. However all of the studies were based on subjective aesthetic responses and used self-reports, which as we explained are open to criticism.

This chapter discusses the design, results and analysis of a novel experiment which was adapted from a path tracing task described in [160], but using a display apparatus design for simulating multiview autostereoscopic displays of varying viewpoint density. The purpose of the experiment is to determine how many views a multiview display might require in order to reproduce acceptable stereo and head motion parallax depth cues when task completion time and accuracy are the dependent variables rather than for aesthetic purposes. Furthermore, the experiment should provide greater understanding of how varying viewpoint densities affects the ability to use head motion parallax as an effective depth

cue; currently this is still unclear in the literature. The results will directly influence the design considerations for our stereoscopic rendering algorithms.

The work contained in this Chapter has been published in ACM Transactions on Applied Perception (TAP), Volume 8, Issue 1, October 2010, “Investigating the performance of path searching tasks in depth on multiview displays”.

3.1 Depth perception and task complexity

In order to better compare stereopsis and motion parallax we attempt to discern how much depth recovery is required for a number of tasks used in past experiments. This will allow us to make a more informed decision on the type of task to use in our experiment. For simplicity we divide the differing possibilities of depth recovery into five levels with each level imposing tighter restrictions on the accuracy of the geometry similar to the hierarchical stratification described by [89] and [17].

- For some tasks it is enough to only be able to detect a difference in disparity or relative motion between two points. The most obvious example is the ability to break camouflage.

- If the sign of disparity or motion differences are recovered (i.e. is an object getting closer to or further away from a reference point) which requires only two views whether from motion parallax or stereopsis, then we are able to perform tasks such as threading a needle [52] or reaching out to touch an object.

- Relative depth judgements or the perception of Bas relief structure can be performed with only two views (either from motion parallax or stereopsis) and relative displacement/velocity of the central point with respect to the others. This restricts the geometry to a set of shapes which can be stretched (affine transformation) along the line of sight and is useful therefore with respect to any geometrical properties that remain constant during these transformations e.g. determining the ratios of depths of parallel line segments (see [150] for a more detailed description). Some tasks that can be completed with this information include being able to distinguish between planar and non-planar objects (between flat and 3D), determine whether an

object is rigid or not, detect shape differences between two objects that can not be made congruent by an affine stretching transformation along the line of sight. Importantly two views and only relative depth judgements are sufficient to determine a unique minimum slant solution which would allow tasks such as path tracing to be performed [89]. Motion parallax can only be used as a depth cue when the motion results in a rotation of an object about an axis which is not the line of sight.

- Shape (scaled) perception with motion parallax can be solved with either a third view [89] or by calculating the angular velocities of corresponding points [41]. Binocular disparity must be scaled with an estimate of the viewing distance [129]. Once the shape is known tasks requiring knowledge of orientation or exact slant become much easier. Motion parallax suffers from depth reversal ambiguities since the sign can not be resolved without the addition of other cues (stereopsis can also suffer from this phenomena in certain cases e.g. inverted sculptures).

- To determine the metric properties of the shape, size and location of an object, the viewing distance must be known and additionally with motion parallax, the angular velocities [41].

It can be concluded that the amount of depth recovery required (and difficulty) varies with the task. The question is what task level is best suited to our purposes in this investigation. Levels 4 and 5 (shape, size and location) are difficult because both cues require extra information e.g. viewing distance or angular velocity estimates. Also the results would be confounded with other variables which would complicate the process of determining the effect on task performance of motion parallax and stereopsis. Tasks in levels 1 and 2 are trivial and unlikely to give interesting results. A large number of problems can be solved with relative depth judgements, therefore any task falling in level 3 would be a sensible choice as the results could easily be applied to many different scenarios.

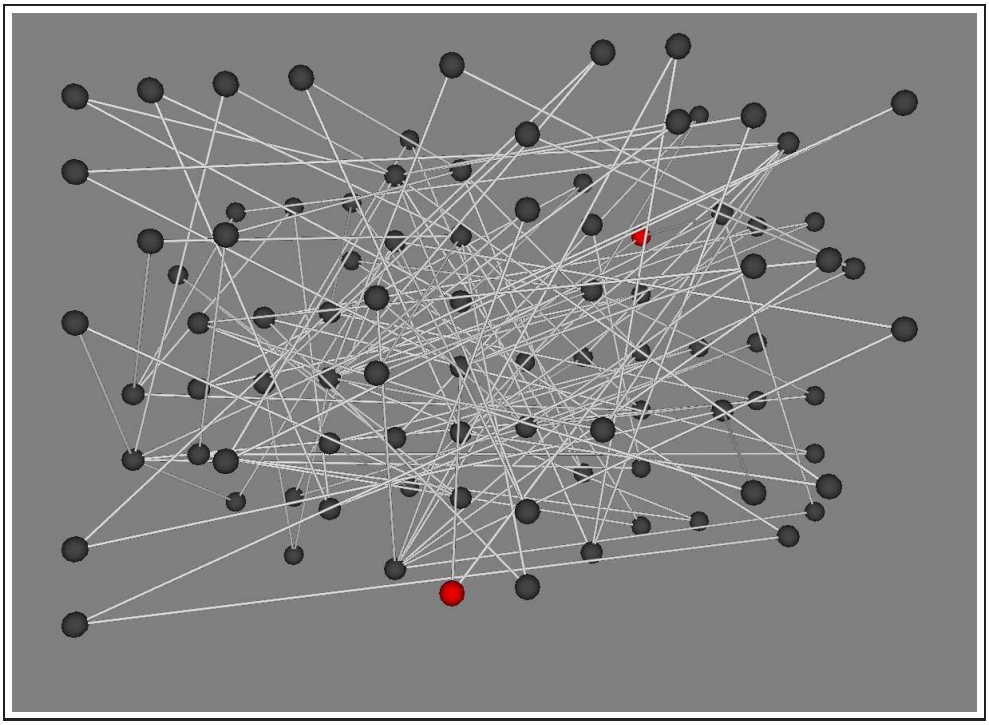
3.2 Experiment 1

3.2.1 Method

Overview

The design used in this experiment was similar to [160] but with some important changes which will be explained shortly. The structure of the graphs and the task remained the same. Participants were presented with a complex of interconnected spheres and asked if they could find a path of two connections (lines/arcs) between two highlighted spheres. Input from the subjects was via the keyboard by pressing ‘y’ or ‘n’ representing yes and no respectively. There was a 50% probability of the two highlighted spheres being connected by a path of two connections. An example of the stimulus is illustrated in Figure 3.1.

Figure 3.1: An example of the stimuli used in this experiment. Subjects were asked whether the two nodes highlighted in red were connected by a path consisting of two arcs. They could only answer yes or no.



For each graph the nodes were divided into three equal-size groups. Two groups contained leaf nodes and the other group, intermediate nodes. Each leaf node was connected

to two different intermediate nodes via cylindrical tubes rather than one pixel width lines as with [160]. Another difference was that each group of nodes were placed into a different depth plane. One group of leaf nodes was set up to reside in front of the screen at some specified distance, while the other group of leaf nodes was placed at an equal distance into the screen, and the intermediate nodes were perceived at the zero parallax plane i.e. the display screen.

The reason for these changes are as follows. There may be depth ambiguities if the thickness of the lines does not change with perspective. Therefore, we chose to avoid any risk of cue conflicts by drawing the lines with cylindrical polygons. Also depth planes were used so that the depth between the two highlighted nodes could be controlled precisely. In [160] the nodes were placed randomly within the volume therefore the depth between any two highlighted nodes is likely to have varied greatly.

Experiment 1 concentrated on determining the effect of viewpoint density with different amounts of stereo depth on the subjects' task performance. We were not interested in varying the complexity of the graphs to quantify how much of an advantage motion parallax and stereopsis can have over the 2D case. Therefore, for this experiment the complexity of all the graphs was kept constant at 90 nodes. Pre-trials indicated this level of complexity was sufficiently difficult that small improvements in depth perception improved task performance. We hypothesized that the experiment would then be sensitive enough to detect any effects small changes in viewpoint density may have.

In previous path searching tasks [160] motion parallax was of more benefit than stereopsis. However it is unclear whether motion parallax was improving depth perception or simply being used to alleviate ambiguities due to occlusion. We tuned our experiment to determine the benefit from motion parallax purely as depth cue. This was achieved by reducing the number of occlusions as much as possible so that from a central viewpoint no nodes occluded each other.

Head-tracking was used to maintain a constant perceived depth and to update the observers virtual viewing position appropriately (motion parallax) by constantly detecting the observer's (x,y,z) position. Head rotation was not tracked but subjects were told to only move horizontally perpendicular to the display. A default eye separation of 6.5 cm was used, although the true eye separation may have varied approximately between 60 and

75 mm [35]; therefore, subjects' perceived depth may also have varied from each other. However, Runde [131] found no advantage in performance when head movements were tracked compared with when they were not tracked and a default eye separation value of 6.5 cm was adopted. We, therefore, assumed that our results would not be affected by the lack of tracking head orientation and eye separation. Stereoscopic camera parameters were calculated using an algorithm described in [83] which allows the perceived depth to be controlled precisely and kept constant without any depth distortions as the observer moves.

There are several reasons for using a path searching task in this investigation. They are relatively simple for subjects to comprehend and do not require much practise time to achieve proficiency, or any previous experience in 3D displays. Also Ware and Franck [160] reported effects on task performance from both motion parallax and stereopsis. Since it is very likely that depth cues will also be of benefit in performing this task, we should be able to study how viewpoint densities affect the viewer's ability to use motion parallax. Furthermore, abstract graphs can be extended to many visualisation tasks on data structures so any results and conclusions based on our results should be applicable to many different scenarios.

Relevant guidelines regarding experimental procedures from [74] were taken into consideration. Furthermore, stereo depth was conservatively kept within a maximum range of 10 cm so as to avoid viewer discomfort [83].

Participants

Thirty-nine subjects were recruited within the University of Durham (12 women, 27 men, mean age 24 years, age range 18-52) and were each paid £5. Participants had normal or corrected-to-normal vision i.e. a visual acuity of at least 20/30 as rated by using a standard Snellen eye chart test and normal stereoscopic acuity (30 sec-arc or better) using the TITMUS test. Participants had varying amounts of experience with 3D displays but were all naive concerning the hypotheses and experimental design.

Apparatus

The computer system used in this experiment was a DELL Workstation PWS360 Intel Pentium 4 CPU 2.26 GHz with 1.00GB of RAM and a NVidia Quadro FX500 graphics card. It was connected to a 19" Hitachi superscan CM813 monitor which was capable of a refresh rate of 120 Hz. Stereo viewing was achieved with a pair of Stereographics Crystal Eyes Workstation glasses with each eye receiving 60 Hz of the update rate resulting in a total frame rate of 60 Hz. The screen resolution was set at 1024x768 pixels. Head tracking was achieved with the InterSense IS-900 Motion Tracking System which consists of a mobile MiniTrax Head Tracker attached to the shutter glasses. The head tracker can accurately detect movements of 0.75 mm within the detectable volume with a latency time of 4ms (as described by the IS900 user manual) and thus a multiview display can be simulated for one observer with a viewpoint density of up to 130 views per 10 cm. The screen brightness was adjusted to the nominal level and the experiment was conducted in a laboratory with minimal light conditions.

Design

Stereoscopic depth and motion parallax from viewpoint density were manipulated as within-subjects (repeated measures) variables in this experiment. Depth had two levels (2 cm and 10 cm[1]) and the viewpoint density had six levels (0, 2, 4, 6, 8, 10 views per 10 cm). For each depth level, the subject would perceive half the scene to come out of the display and the other half into or behind the display.

A viewpoint density of 0 views per 10 cm represents the case for no motion parallax; only the standard two views required for stereopsis are produced from a fixed viewpoint. A control was also set-up on the display which produced no depth and no motion parallax reflecting the standard 2D monitor case. Each subject was required to repeat the task six times for each condition resulting in 78 trials and 78 randomly generated graphs (items). The order in which subjects performed the task on each item with the different treatments

[1]We do not describe the depth using angular disparities because with a fixed angular disparity the perceived depth still varies with viewing distance. Also, we have implemented a camera control method to maintain a constant perceived depth regardless of the observer's head position (see [83] for more details).

was counterbalanced and followed a Latin Square design. Each item was presented to the subject in an equally random fashion.

Procedure

Experimental sessions lasted from 45-60 minutes. On arrival at the laboratory, subjects were screened for normal vision using a TITMUS and Snellen eye chart test. If they passed the initial screening test they were given written instructions on how to perform the task before them. The subjects were seated at a distance of 65 cm from the display. Accuracy was emphasised over speed with the motivation of winning a small prize for whoever achieved the most correct answers. In order that fatigue did not set in, short breaks were allowed after every twenty answers given. Prior to the start of the experiment subjects were required to practice the task on graphs which were randomly generated on the fly so as to become proficient in the task. This practice session was informal and continued until the subjects were confident that they understood the task. Subjects were required to wear the shutter glasses for all the trials regardless of whether there was any depth present in the scene. They were also encouraged to move their heads as much as possible to gain any possible advantage from motion parallax. Upon completion of the experiment, participants were debriefed and given the opportunity to ask any questions. Also the subjects were asked to rate their tolerances on a scale of 1 to 5 of the flipping effect and false rotation effect caused by low viewpoint densities and any visual fatigue or discomfort experienced during the experiment via a questionnaire (where 5 indicates a high degree of intolerance and also extreme visual fatigue or discomfort).

3.2.2 Hypothesis

Explorative tasks of abstract data structures can be very difficult in 2D especially when there are large amounts of data. The main obstacles to correctly interpreting the data are ambiguities due to arc crossings, occlusions and lack of spatial awareness i.e. everything is perceived to lie on a single plane. Improving depth perception with either motion parallax or stereopsis should improve task performance. For example, slant information which can be obtained from relative depth judgements, allows the observer to more readily trace the path of the arcs [89].

Previous investigations using path searching tasks found depth perceived from motion parallax and binocular disparity increased accuracy of the responses, decreased the response latencies and observed the greatest performance gains when both cues were present simultaneously [160, 162]; we expect similar results.

A significant difference between our experiment and [160,162] is that motion parallax is perceived from visibly discrete viewpoints. Phenomena such as the flipping effect and false rotation effect become apparent at low viewpoint densities and we suspect this will hinder the observer in judging depth from motion parallax. We predict that increasing the viewpoint density will improve accuracy and response latencies.

3.2.3 Results

We subjected the data to analyses of variance (ANOVAs) for the subjects (F_1) and for the items (F_2) with depth and viewpoint density as within-subjects independent variables and response latencies and percentage of correct responses as the dependent variables. Performance was generally good indicating all the subjects understood the task well (average percent of correct responses was 90% and everyone scored at least 72%).

	Depth 2 cm		Depth 5 cm		Control	
Viewpoint Density	M	SD	M	SD	M	SD
0	18.94	8.58	18.22	7.45	19.58	9.80
2	18.01	8.32	17.95	7.81		
4	17.80	9.09	17.07	7.70		
6	18.27	8.21	16.15	5.87		
8	18.47	7.57	16.19	6.84		
10	18.30	6.48	16.77	7.93		

Table 3.1: Average response latencies (in secs) of the subjects under all the conditions in Experiment 1.

Response Latencies

Table 3.1 shows the mean value and standard deviation of the response latencies for each experimental condition. Only the correct responses were taken into account when analysing the times.

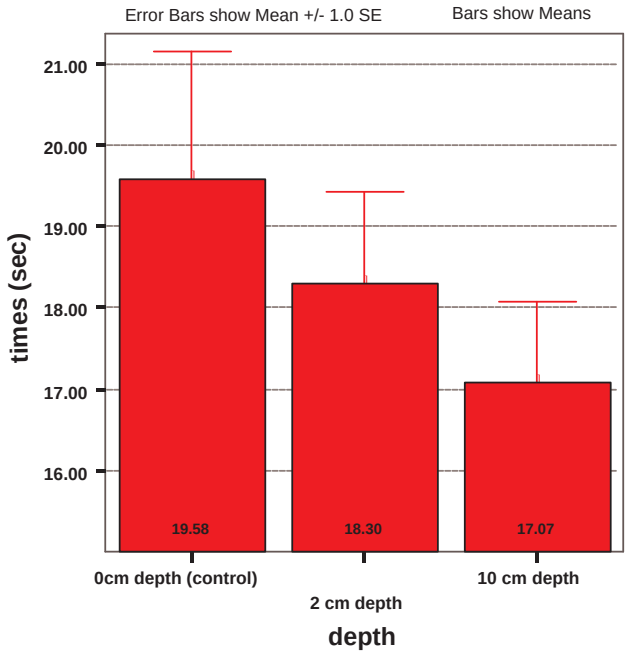


Figure 3.2: Mean response latencies from the subjects under three depth-cue conditions collapsed across the viewpoint density variable in Experiment 1.

We expected that increasing the viewpoint density would improve the performance; however, a repeated-measures analysis of variance revealed that the viewpoint density had no effect on the response latencies, $F_1(5, 190) = 1.17$, $p = 0.33$, $F_2(5,385) = 1.17$, $p = 0.32$ and there was no interaction between the depth and viewpoint density, $F_1(5, 190) = 0.86$, $p = 0.51$, $F_2(5,385) = 0.41$, $p = 0.85$. On the other hand, the amount of perceived depth from stereopsis had a significant effect on the response latencies, $F_1(1, 38) = 9.20$, $p < 0.01$, $F_2(1,77) = 10.48$, $p < 0.01$.

Since no effect was found for varying the number of views, the data were collapsed across the viewpoint density variable. Figure 3.2 shows that the response latencies decreased as the amount of depth increased, which is in agreement with our prediction.

However, there was no significant difference between the control and the 2 cm depth condition, $t_1(38) = 1.40$, $p = 0.08$, one-tailed, $t_2(77) = 0.99$, $p = 0.16$, one-tailed, but there was a significant difference between 10 cm and 2 cm of depth, $t_1(38) = 3.03$, $p < 0.01$, one-tailed, $t_2(77) = 3.90$, $p < 0.01$, one-tailed.

	Depth 2 cm		Depth 5 cm		Control	
Views per 10cm	M	SD	M	SD	M	SD
0	86.75	13.34	89.74	11.86	77.35	17.31
2	88.46	12.77	92.31	10.71		
4	88.46	14.39	92.31	12.00		
6	89.74	12.46	93.59	11.22		
8	90.60	13.13	94.87	10.92		
10	88.89	13.96	93.59	13.03		

Table 3.2: Percentage of correct responses for the subjects under all the conditions in Experiment 1.

Percentage of correct responses

Table 3.2 shows the mean value and standard deviation of the percentage of correct responses for each experimental condition.

A repeated-measures analysis of variance revealed viewpoint density had no effect on the percentage of correct responses either, $F_1(5, 190) = 1.42$, $p = 0.22$, $F_2(5,385) = 1.63$, $p = 0.15$ and no interaction between depth and viewpoint density, $F_1(5, 190) = 0.06$, $p = 1.00$, $F_2(5,385) = 0.06$, $p = 1.00$. In contrast, the stereo depth had a significant effect on the percentage of correct responses, $F_1(1, 38) = 12.31$, $p < 0.01$, $F_2(1,77) = 5.99$, $p < 0.05$.

As before the data were collapsed across the viewpoint density variable. Figure 3.3 shows that as depth increased the percentage of correct responses also increased, which was expected. There was a significant difference between the control and the 2 cm depth setting, $t_1(38) = -4.20$, $p < 0.01$, one-tailed, $t_2(77) = -3.43$, $p < 0.01$, one-tailed and the 10

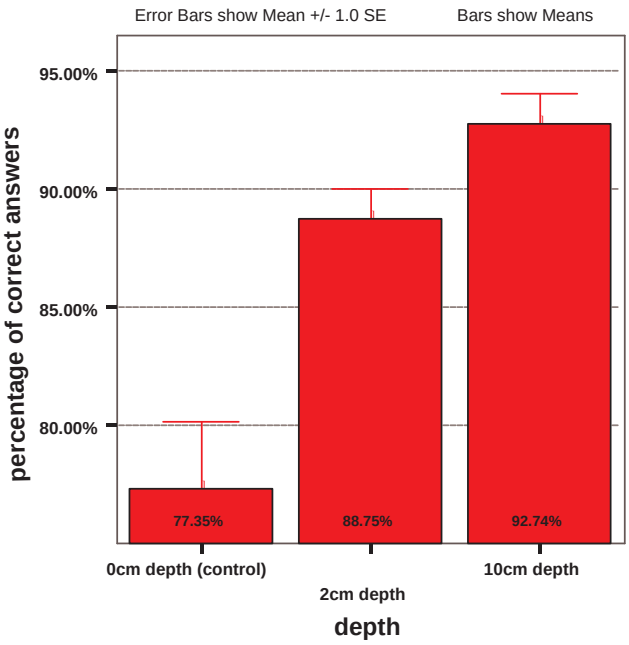


Figure 3.3: Mean percentage of correct responses from the subjects under three depth-cue conditions collapsed across the viewpoint density variable in Experiment 1.

cm setting showed a significant difference from the 2 cm setting as well, $t_1(38) = -3.47$, $p < 0.01$, one-tailed, $t_2(77) = -2.49$, $p < 0.01$, one-tailed.

Subjective responses

The mean subjective response for tolerance of low viewpoint densities was 2.28 with a standard deviation of 0.86 and the mean subjective response for visual discomfort was 1.85 with a standard deviation of 0.93. These results suggest that the majority of subjects did not suffer much visual discomfort and were not greatly dissatisfied with low viewpoint densities. Furthermore, there were no comments either written or verbal made to the authors regarding any problems with low viewpoint densities.

Head movement range

A potential problem with our experiment was that the observers were not forced to make head movements. Figure 3.4 presents a box plot showing the maximum lateral distance

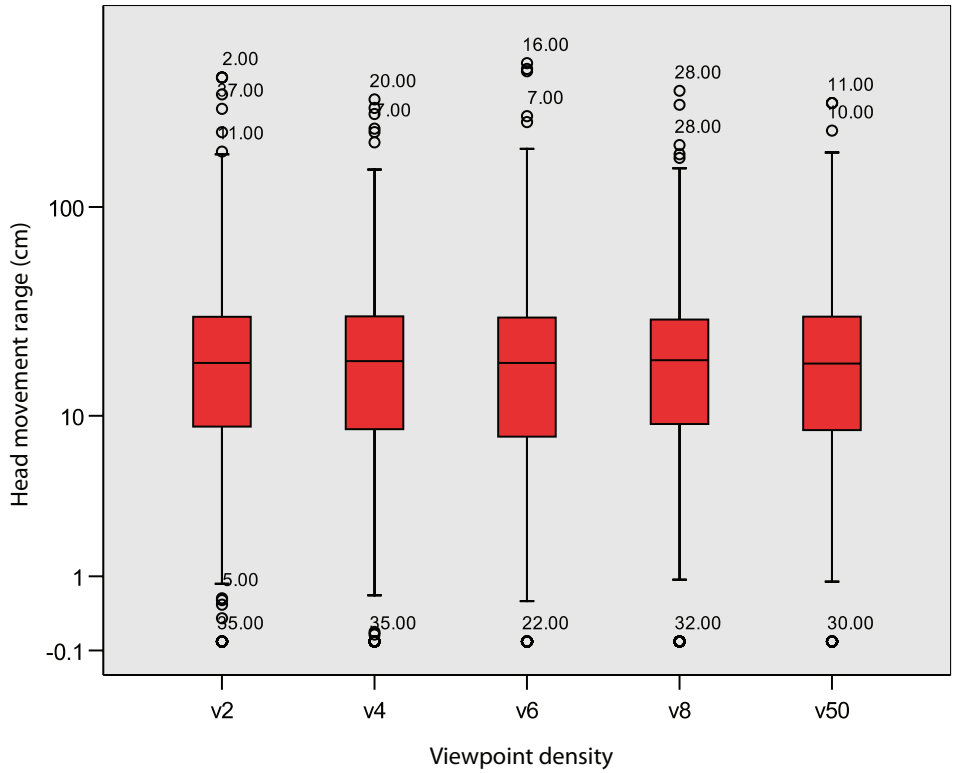


Figure 3.4: Box plot showing the maximum lateral distance subjects moved their heads by in Experiment 1.

moved by all the subjects for the cases where motion parallax was available. While a number of results showed no or little head movement, these are considered outliers and the vast majority of the results showed a satisfactory range of head movement; on average the subjects moved more than 24 cm. While subjects were reminded to stay seated during the trial, occasionally they did lean out of the chair, as can be seen in the data by head movement ranges greater than one metre.

Discussion

From the results we can make a firm conclusion that depth perception affects task performance in path searching tasks which is in agreement with the literature. Even gaining only a small amount of stereo depth (2 cm) dramatically improved task performance regarding the percentage of correct responses (increase of 11.4%). Larger amounts of depth

only improved performance by a further 4%. These findings help support the case for using microstereopsis rather than larger disparities.

Siegel et al. [140] define microstereopsis as stereo viewing with a camera base separation of about 3% of the interocular distance. They suggest from a series of informal experiments that good depth perception is still possible with microstereopsis when lots of other strong monocular depth cues are present. Unfortunately the scene depth and disparities were not specified so the exact definition of microstereopsis is unclear; however, a gray level representation of the differences between the stereo image pair appears to contain a maximum disparity of only a few pixels. For the 2 cm case in our experiment the maximum disparity was approximately 3 pixels and therefore we consider it as an example of microstereopsis.

Visual comfort of stereo viewing is affected by many factors, including optical properties of the display, display size, viewing distance, stereo camera capturing methods and even the scene content [95]. With typical desktop viewing conditions (viewing distance of 65 cm) depth greater than 24 min of arc, which only relates to 4.89 cm behind the display and 4.25 cm in front, can cause fatigue and visual discomfort [83]. Too much depth for long durations can be very taxing on the visual system [110, 111, 120] and could possibly decrease task performance. Therefore, if a person was expected to use a 3D display for long periods of time, depths as little as 2 cm could be used while still gaining a significant advantage and minimising visual discomfort.

Stereo depth as little as 2 cm can easily be perceived (as indicated by the threshold sensitivities of the TITMUS test) and although reduced errors, did not significantly affect response latencies. However, 10 cm of stereo depth did improve timings. For tasks which can be solved with only relative depth judgements, two different magnitudes of disparity should theoretically impart the same amount of useful information since the ratio of depths of features remains constant [81]. The task may have taken longer with less depth simply because it was harder or possibly the visual processes responsible for binocular fusion work slower with smaller amounts of disparity.

3.3 Experiment 2

In the previous experiment motion parallax was found to not have any additive effect with stereopsis on task performance. Findings from previous path searching investigations, however, strongly suggest motion parallax should significantly improve task performance. Furthermore, many investigations [17, 71, 116, 130] have shown head motion parallax improves depth perception so we can assume in the absence of significant occlusion, head motion parallax by itself should still improve task performance for our modified path searching task. The previous experiment was an attempt to determine the required number of views under natural viewing conditions for multiview displays since head motion parallax is always coupled with binocular disparity. Experiment 2 is essentially a repeat of the first experiment but with the stereo cue switched off so as to isolate the effect head motion parallax with varying viewpoint densities has on task performance.

Participants

Fifteen candidates were recruited within the University of Durham (4 women, 11 men, mean age 24 years, age range 21-31) and were each paid £5. Participants had normal or corrected-to-normal vision i.e. a visual acuity of at least 20/30 as rated by using a standard Snellen eye chart test. Participants had a varying amount of experience with 3D displays but were all naive concerning the hypotheses and experimental design.

Stimuli, design, apparatus and design

The task and apparatus were exactly the same as in the first experiment. Viewpoint density was manipulated as a within-subjects (repeated measures) variable with five levels, (2, 4, 6, 8, 50 views per 10 cm). Subjects repeated the task 20 times for each level and were presented in total with 100 graphs. The camera model was set-up identically to the 10 cm condition in the previous experiment but the observer only received the left view in both eyes, i.e. a monoscopic image. Head position was only tracked laterally and observers were strongly encouraged to only make lateral head movements. The procedure then remained identical to that in Experiment 1.

3.3.1 Results

The data were again subjected to analyses of variance (ANOVAs) for the subjects (F_1) and for the items (F_2) with viewpoint density as a within-subjects independent variable and response latencies and percentage of correct responses as the dependent variables. Performance was slightly worse than in the previous experiment but still generally good with an average percentage of correct responses of 89% and everyone scored at least 65%.

	Depth 10 cm	
Views per 10 cm	M	SD
Subjects		
2	19.49	3.69
4	16.70	6.08
6	17.44	7.08
8	15.22	5.03
50	16.42	5.52

Table 3.3: Average response latencies (in secs) for the subjects under all the conditions in Experiment 2.

Response latencies

Table 3.3 shows the mean value and standard deviation of the response latencies for each experimental condition. Only the correct responses were taken into account when analysing the times.

A repeated-measures analysis of variance indicates viewpoint density had a significant effect on response latencies, $F_1(4, 56) = 3.69$, $p < 0.05$, $F_2(4,396) = 8.86$, $p < 0.01$. Latencies were shortest for 8 views and longest for 2 views; a means comparison between the 2 views condition and more views showed that the majority of differences were reliable (for the t_1 comparisons the pairs 2 vs. 4, 2 vs. 8 and 2 vs. 50, and for the t_2 comparisons 2 vs. 6, 2 vs. 8, and 2 vs. 50 were all significant; all ts > 2.33, ps $<= 0.05$). Other differences were not significant.

	Depth 10 cm	
Views per 10 cm	M	SD
Subjects		
2	87.67	14.00
4	87.67	10.83
6	88.33	12.20
8	90.33	9.72
50	91.00	11.37

Table 3.4: Average percentage of correct responses for the subjects under all the conditions in Experiment 2.

Percentage of correct responses

Table 3.4 shows the mean value and standard deviation of the percentage of correct responses for each experimental condition. Although there appeared to be a trend of improved performance as the number of views increased, a repeated-measures analysis of variance indicated viewpoint density had no effect on the percentage of correct responses, $F_1(4, 56) = 0.53$, $p = 0.72$, $F_2(4,396) = 0.94$, $p = 0.44$.

Head movement range

Figure 3.5 presents a box plot showing the maximum lateral distance moved by all the subjects in Experiment 2. Again there were a number of results which showed no or little head movement; however, in this case they are considered extreme outliers. The subjects moved on average 27 cm.

Discussion

Motion parallax under monoscopic (2D) viewing conditions did affect task performance in this experiment, but the effect was much weaker than with stereopsis and viewpoint densities only affected response latencies. The shortest response latencies were observed with 8 views; increasing the view point density by a relatively large amount to 50 views

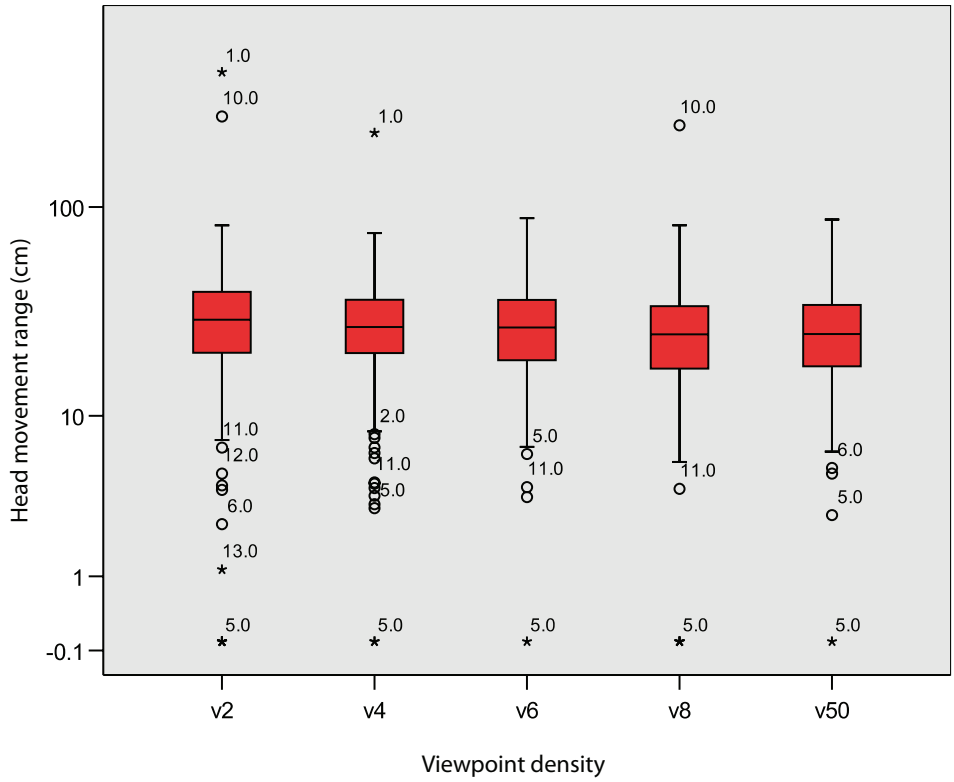


Figure 3.5: Box plot showing the maximum lateral distance subjects moved their heads by in Experiment 2.

did not significantly improve the subjects' performance. While the data was noisy, this suggests that the maximum benefit from motion parallax in a path searching task with limited occlusion can be achieved with 8 views. Also while not statistically significant Table 3.4 does appear to show a trend of improved response accuracy with greater viewpoint densities. It is possible that only a proportion of the subjects benefited from the increased number of views for a proportion of the stimuli, i.e. there was some effect but it was not reliable.

Sollenberger and Milgram [142] concluded depth perception from motion parallax improved mostly from continuous rotation and not through discrete viewpoints which reflects our findings that response latencies significantly decreased with increased number of views. The average response latency decreased by over 21% from 2 views to 8 views. There was not a significant difference between the response latencies from 8 views to 50

views even though the increase in viewpoint density was large. This suggests 8 views per 10 cm or a view every 1.25 cm is the upper limit on the required number of views to improve task performance. The results could also suggest that only 4 views are actually needed since there was no significant difference between 4 views and 50 views either; however, further research may be required to support such an aggressive restriction on viewpoint density.

The average head movement range appeared to be larger in the second experiment (at least 3 cm more for all the viewing conditions). This suggest, in the absence of stereo, subjects make more effort to take advantage of the motion parallax available.

3.4 General Discussion

The general consensus in the literature is that the visual system combines information from all the available depth cues to create the most accurate model of depth from the scene as possible. Various models have been derived both statistically and by observation. A popular model is the weighted linear combination strategy, see for example [18, 20, 40, 82, 96, 177]. Also depth cues may not be entirely uncorrelated since they are likely to share noise sources from the retina and affect some of the same neural mechanisms e.g. disparity and motion parallax both affect neurons in the cortical area MT [117], which further supports the theory of cue combination. However, the visual system may combine any number of available cues immediately, dynamically adjust their reliability or ignore some in favour of others [97]. Therefore interactive effects between the cues can not always be expected.

An important consideration to take into account is the task itself. The benefit of motion parallax and stereopsis depends greatly on the task, stimulus and experimental procedure as shown via a number of different tasks with identical viewing conditions [17, 52]. Merritt and Cole [109] concluded when stereopsis was available, potential depth information from motion parallax was not always used.

Therefore, the fact that an additive effect on task performance with head motion parallax and stereopsis was not observed is not necessarily surprising. However, a few important conclusions can be derived from the differences in our results to [160]. Firstly, we

demonstrated in Experiment 2 that although task performance in our experiment could be improved by motion parallax, the advantage was not as great as from stereopsis. The most likely explanation for this difference is due to the reduction in occlusion. Secondly our results suggest either one of these two possibilities:

- Motion parallax does not actually have an additive effect with stereopsis on depth perception. Rather it may have an additive effect on task performance if there is a significant amount of occlusion in the scene.

- In the stimuli we rendered the connections between the nodes using large well shaded cylinders as opposed to single pixel lines with no perspective as in [160] or with very thin tubes with no shading in [162]. Motion parallax has been suggested to have an additive effect on stereopsis because it may help with the correspondence problem [149]. However, if there are other cues such as perspective and shading, the correspondence problem is less difficult and motion parallax is less effective.

3.5 Conclusion

We conducted two experiments which investigated subjects' performance in a path searching task with varying amounts of stereo depth and viewpoint densities. Performance significantly improved with greater amounts of stereo depth. Accuracy improved even with depths as little as 2 cm. Excessive disparities therefore can be avoided, as applications may still benefit from small amounts of stereopsis while also keeping visual discomfort due to accommodation and vergence mismatch at a minimum.

Generally as the viewpoint density increased so did the advantage gained from head motion parallax. However, viewpoint densities greater than 8 views per 10 cm did not improve task performance any further. This suggests the upper limit required on the viewpoint density is quite low. Furthermore, when stereopsis was available head motion parallax did not have any effect on task performance. Therefore, for certain applications, e.g. path searching tasks where occlusion is not an overiding factor, the main advantage gained from multiview displays over two-view stereoscopic disays is viewing freedom. In these cases, our results suggest the optimum multiview display design is that of low viewpoint densities with the views repeated across several viewing lobes.

Chapter 4

Rendering multiple views with controllable depth using an incremental fragment algorithm for particle data sets

A number of rendering algorithms for 3D displays have been discussed in the background chapter; however, none have been specifically designed with uncorrelated particle data sets, such as cosmological N-body and SPH simulation output, in mind, which is a growing problem. This chapter explores the potential for saving computation in the rendering pipeline by perspective reprojection. While stereoscopic algorithms employing perspective reprojection have been described before, the novelty in our approach is in applying the technique solely to point geometry and delaying reprojection and generation of the viewpoints until the fragment stage of the pipeline. Efficiencies can be realised by then eliminating redundant stages such as occlusion handling and reusing all the calculations prior to the fragment stage across the viewpoints. This also greatly improves performance for large data sets, since the data need only be sent to the graphics card once regardless of the number of viewpoints. We discuss implementation details and issues, and demonstrate the improved performance of our algorithm compared to a conventional rendering pipeline.

4.1 Motivation

Currently the most suitable method for rendering multiple views from uncorrelated 3D particle data sets is a point-based rendering algorithm described in [76]. However, a significant proportion of the work-load in all PBR algorithms involves surface reconstruction, which is of no benefit when rendering uncorrelated 3D points. Further optimisations can be made by assuming all the particles are spherical and translucent. This means an additive blending technique can be used, eliminating the requirement for dealing with surface normals, depth sorting and occlusion handling. Incorporating these optimisations with an algorithm which also takes advantage of the available stereo-coherence between the views should result in a much more efficient multiview rendering algorithm for uncorrelated particle data sets.

Three important observations were taken into consideration when designing our algorithm: firstly, our experiments described in chapter 3 have shown humans are surprisingly insensitive to the number of viewpoints with regard to their task performance, suggesting only a few views are required for viewing freedom. Therefore, multiview rendering algorithms must offer immediate performance increases for more than one view. In other words, initial set up costs and rendering of two views should ideally not be greater than the time taken to render those views with conventional single view rendering algorithms.

Secondly, the following geometric similarities between stereoscopic viewpoints (assuming a right-handed coordinate system) are available:

- A projected point onto the projection plane has exactly the same y-coordinate for all the different viewpoints.
- It also follows that all the points with the same depth and vertical position will have identical y-coordinate positions on the projection plane and for all the other viewpoints too.
- Assuming the distance between each camera is constant, then the disparity of a projected point from one consecutive viewpoint to another is also constant (for proof see [24, 118]).

Thirdly, an issue with stereoscopic image generation is that without careful consider-

ation of the camera separation, depth distortions can occur as the viewer moves laterally to look around the scene, and excessive parallax can be captured causing viewer discomfort [83]. Adjusting the camera separation manually can be a tedious trial and error affair; therefore, our algorithm must be able to support a suitable depth control mechanism. For this reason we chose to implement a parallel camera model which eliminates distortions such as keystone and allows the scene volume to be mapped onto an arbitrary perceived volume quite conveniently by using the equations described in [83].

Castle [24] describes an algorithm which improves the rasterisation performance by incrementing the projection of each vertex by a fixed amount for each consecutive view and interpolating the intersection of the polygon edges with each scan-line. Our algorithm takes this one step further by rasterising each point/particle only once and then incrementing the pixels associated with that point by the appropriate disparity. This has the benefit that all the previous calculations involved in rendering, e.g. transformations, projection, lighting and rasterisation need only be performed once regardless of the number of views that require rendering.

4.2 Problem description and rendering method

As mentioned before, the primary problem this thesis addresses is the lack of efficient stereoscopic rendering for uncorrelated 3D point data sets. For the development and evaluation of our algorithm we obtained our data by cutting a 22 Mpc/h sphere from the Millennium Simulation [145] and consisted of a number of different time snapshots ranging from a redshift of 15, in which the universe was 16 times smaller in each of the three dimensions than today, to 0 which is the present day. The Millennium Simulation was produced using GADGET [144, 146] which is a very popular particle simulation software package capable of performing cosmological N-body and SPH simulations on massively parallel computers; it computes gravitational forces with a hierarchical tree algorithm and represents fluids by means of smoothed particle hydrodynamics (SPH).

Hopf et al. [68, 69] describe a suitable approach to rendering GADGET data using vertex shaders to splat the particles with either point sprites or OpenGL anti-aliased points which approximate the splats. We adopt this approach of approximating the splats and

using vertex shaders to improve the rendering performance (the vertex shader we used can be found in the Appendix A.1).

The GADGET output was processed so that the data files only contained the position (x,y,z coordinates), density and temperature for each particle. The opacity of each particle is calculated with a user defined transfer function which also takes into account the density of the particle. A sigmoid function describes an S-shaped curve and gives good results as it can be used to gradually increase the opacity of denser particles and reveal the structures within the volume of data. The following sigmoid function was used:

$$\alpha = \frac{1}{1 + e^{-\frac{1}{3}(\rho + s)}}$$

Where α is the opacity to be calculated, ρ is the density of the particle and s is an arbitrary number used to shift the function left or right so as the overall appearance of the scene can be adjusted. Denser particles are brighter and contribute more of their colour to the output pixels than less dense particles.

The size of the particles after projection are scaled according to their density using a smoothing function. The same sigmoid function can be used again so as to save on the number of calculations required, and then the values are scaled into an appropriate range for the different particle sizes. The combination of a transfer and smoothing function can give results such as smooth cloud like renderings, typical of volume rendering, or more detailed renderings of the structure.

4.2.1 Blending, lighting and occlusion

Blending is very useful in rendering large particle data sets, since interesting structures can be observed within the global mass of particles. Different blending functions will give different results, so it is important to choose the appropriate function for the type of data to be rendered. The most common way to blend, especially in volume rendering applications, is to to sort the particles in depth order and render the furthest away from the camera first. To blend a foreground pixel onto the background, involves multiplying the colour of the background by one minus the alpha value of the foreground pixel and adding it to the colour the foreground pixel multiplied by its own alpha value. The result is similar to using colour filters and having the scene lit up from behind. This type of

blending allows dense regions of particles to show up nicely, with their correct colour within the entire volume of particles. However, performing a depth sort on millions or even billions of particles is not a trivial task.

If we assume each particle emits light and is translucent at the same time, then cumulative blending, which does not require a depth sort, can be used. OpenGL uses the following blending function (as described by the OpenGL specification version 2.1):

$$R_sS_r + R_dD_r, G_sS_g + G_dD_g, B_sS_b + B_dD_b, A_sS_a + A_dD_a$$

Where the s and d subscripts are the source and destination pixels respectively and the S and D components are the blend factors. Cumulative blending involves multiplying the foreground pixel by its own alpha value and adding it to the background pixel by assigning S and D as (As, As, As, As) and $(1,1,1,1)$ respectively, resulting in the following function:

$$R_sA_s + R_d, G_sA_s + G_d, B_sA_s + B_d, A_sA_s + A_d$$

This type of blending is often used in games for explosions and clouds, but is also suitable in volume or particle rendering [69].

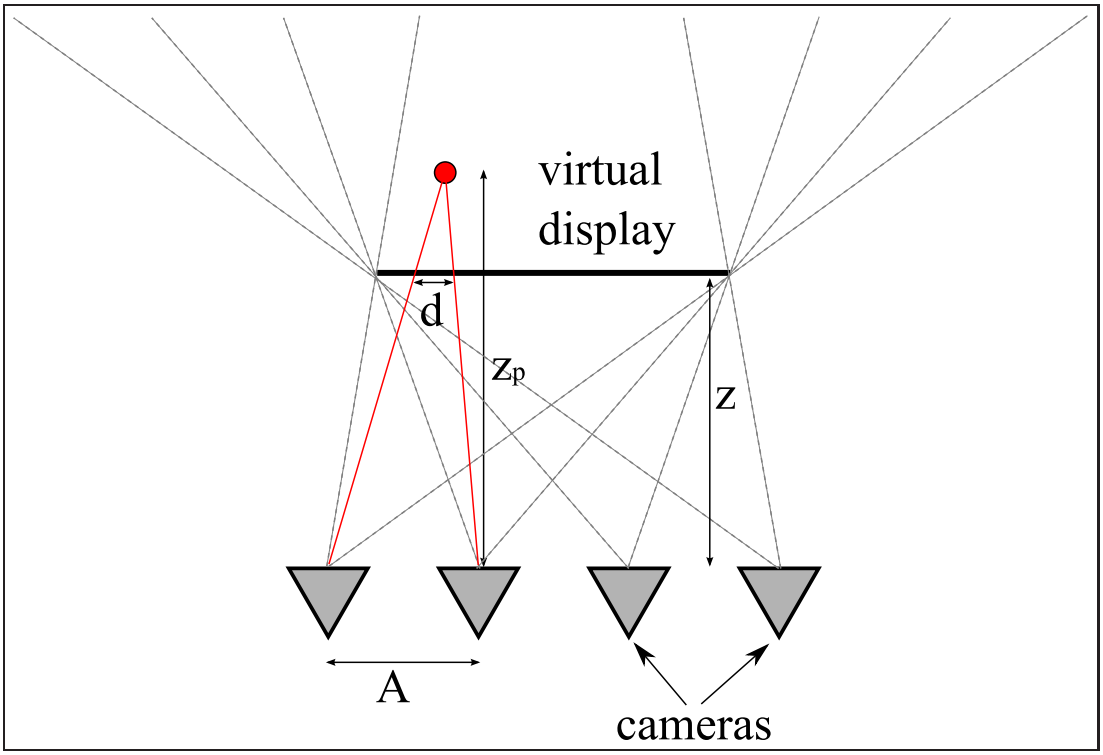
The colour of each particle was calculated from its temperature using the mix() function available in OpenGL's shader language. The coolest and hottest temperatures are given user defined colours; therefore, every particle's colour varied between the two hues depending on its temperature. Occlusion and multiview visibility determination are not taken into consideration since every particle is translucent, and cumulative blending is performed; therefore, every particle contributes some of its colour.

4.3 An incremental fragment algorithm

Since the particles represent spheres with no surface normal, the splats remain the same shape regardless of the viewpoint. Also, lighting, splat size and opacity for each particle are constant across the parallel views; therefore, upon examining the rendering pipeline, it becomes apparent only the last stage after rasterisation, which involves alpha blending and updating the frame buffer, needs to be changed to render multiple views and all the calculations prior to this stage can always be shared across the views. The particles are

projected, splatted and rasterised as described in the previous sections for a single view. The extra views can be calculated from this initial rendered view in the fragment stage of the pipeline.

Figure 4.1: Geometry of multiple cameras.



The disparity for the projected points between consecutive views behind the display is calculated with the following equation:

$$d = \frac{A(z_p - z)}{z_p}$$

Where d is the disparity, z_p is the depth component of the point to the camera, a is the multiview camera separation between each consecutive viewpoint and z is the viewing depth of the camera to the virtual display (see Figure 4.1). Whereas, disparity for points in front of the display are calculated with a slightly different formula:

$$d = \frac{A(z - z_p)}{z_p}$$

We noted earlier, for any particular depth, the disparity between corresponding points in the consecutive views is constant if we assume the camera separation is also constant

and the vertical projection position does not change across the views. Therefore, the disparity for each point need only be calculated once, and the fragments representing that point can be shifted horizontally and in an incremental fashion by a constant amount for each consecutive view. Figure 4.2 presents the pseudocode for the incremental reprojection stage after rasterisation.

Figure 4.2: Pseudocode for the incremental reprojection fragment algortihm

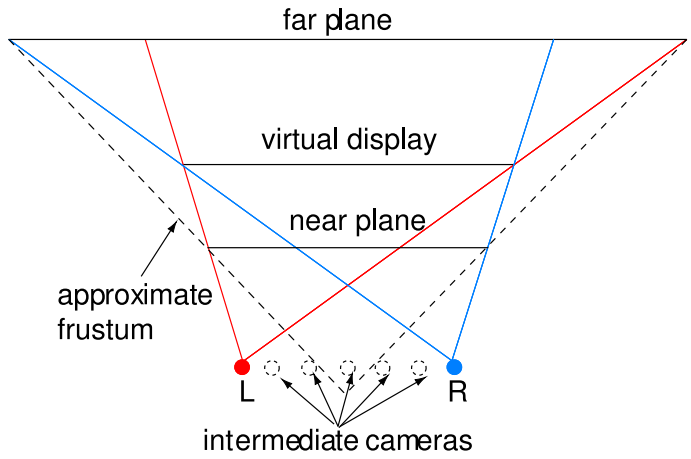
```
    procedure incrementFragments {
(1)   for each point {
(2)     calculate disparity
(3)     for each view {
(4)       increment the pixel position
(5)       if pixel visible update frame buffer
        } next view
      } next point
    } end incrementFragments
```

4.3.1 Viewing frustum

Since the points are only being projected once, regardless of how many views are required, the initial viewing frustum must be large enough to encompass all the visible points from the left and right extreme views as shown in Figure 4.3. However, the shape of the viewing volume complicates culling. One solution described by Castle [24] is to set up an approximate viewing frustum which roughly discards most of the points that are definitely not visible in any of the views, and then after projection and calculating the position of the points for all the views, another culling process is carried out for each view.

However, it is desirable to only project points which will be visible in the viewpoints so as to reduce inefficiency. In order to do this, two separate frustums must be set-up for points behind the viewing display and points in front (see Figure 4.3). This also allows

Figure 4.3: Viewing frustum of left and right extreme views. The approximate frustum is large enough to encompass all the points visible from all the different viewpoints.

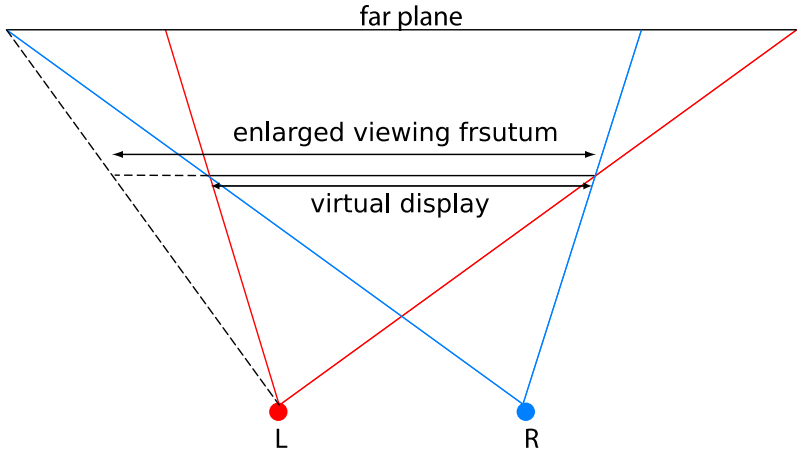


a slightly different shader program to be used for each frustum since the incremental method calculates positive and negative disparity with different equations.

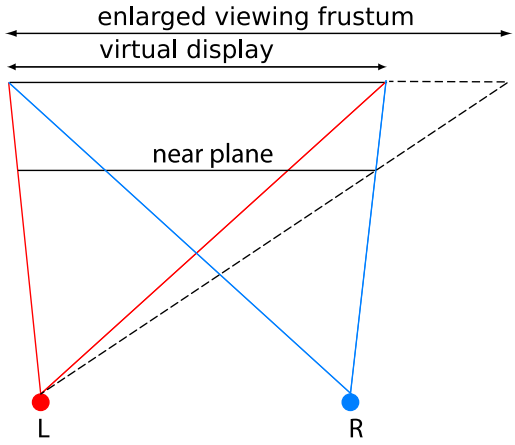
The two frustums are set-up for either the left-most or right-most view and the extra views are rendered incrementally. We consider the case for rendering the left-most view. Figure 4.4 shows the two frustums required for points with positive and negative disparity, which are large enough to encompass all the points visible by all the views. There are two methods to ensure that each viewpoint is displayed with the correct incremented points. The first method is to determine whether each incremented point is within the horizontal viewing window boundaries for each view before updating the associated frame buffer for that view. If we assume the entire scene must be visible in each view, as it was in our case, the incremented pixel positions can be directly updated into each frame buffer without any boundary checks.

4.3.2 Controlling the perceived depth

The camera method [83] can be tuned to each observer's eye separation to maintain a constant perceived depth. However, this only works correctly for either two-view displays with the viewer positioned centrally, or when head tracking is incorporated into the display. Either way, only one observer can be supported for correct depth perception.



(a) Positive disparity frustum



(b) Negative disparity frustum

Figure 4.4: The original left-most viewing frustums for points with either positive or negative disparity are enlarged so as to cover the volume visible across all the viewpoints. This allows all the views to be rendered in a single pass by reprojecting the points in the fragment stage of the pipeline.

Multiview displays usually have fixed viewing windows in viewer space and so the parallax should be tuned for each viewing window rather than an individual's eye separation. We can adapt the camera model described by [83], by replacing the eye separation with the viewing window width for the specified viewing distance.

GPU limitations

For the greatest efficiency it is desirable to implement the algorithm on the graphics card, for example by using shaders. However, currently the programmable model of pixel

shaders is very limited and does not allow array indexing of the pixel positions in the fragment stage, which means the U,V coordinates of each fragment cannot be changed. Unless this feature is added in future generations of GPUs our algorithm can only be implemented using the CPU.

Another issue is that although each point is represented by a group of fragments, currently fragment processors can only operate on one fragment at a time and have no access to neighbouring fragments. Therefore, each pixel/fragment represented by a particular point must have its disparity calculated and reprojected individually. Obviously, it would be far more efficient to calculate the disparity for one point, and increment all the fragments representing that point at the same time. Fragment clustering, i.e. sending multiple fragments to the fragment/pixel shader at the same time would be another desired feature.

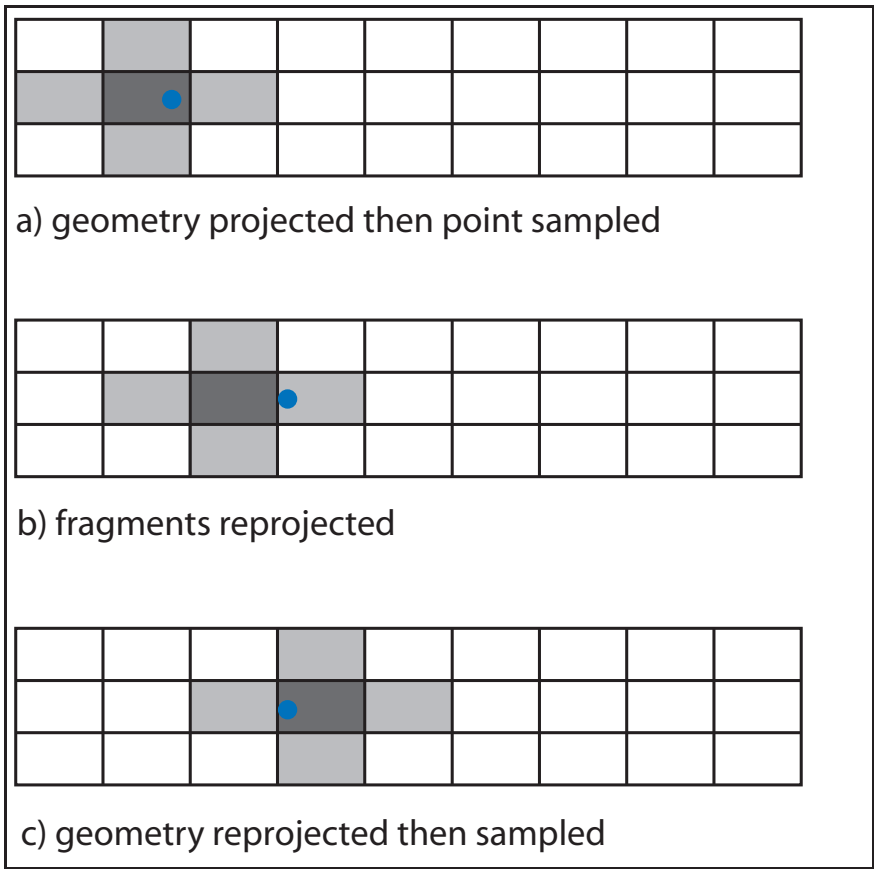
Sampling issues

To recap, the following aliasing effects [125] can occur in stereoscopic rendering: inaccuracies in projected position, inaccuracies in projected size, inaccuracies in disparity, inconsistencies in projected size, inconsistencies in disparity, inconsistencies in disparity of horizontal edges and inconsistencies in position. Inconsistencies of the projected size can only occur when an object is represented by two or more endpoints. However, since each splat is rendered using either a point sprite or a smooth OpenGL point, this problem is eliminated. This also has the effect of eliminating inconsistent disparities of the horizontal edges. Further inconsistencies in disparity are eliminated with our algorithm as the fragments are reprojected by a constant amount.

Unfortunately, certain geometrical distortions can occur when the geometry is sampled first and then the sampled pixels reprojected [24]. Reprojecting can lead to a pixel difference from correctly sampling the geometry as illustrated in Figure 4.5 when a point has a disparity of 1.2 pixels two corresponding views. Furthermore, the error in sampling can get progressively worse in an incremental algorithm if the problem is not addressed carefully. For example, if the disparity of a fragment is rounded down from 1.2 to 1 pixel and reprojected by this amount, then after five views there may be up to two pixel difference from the correctly sampled image.

An obvious solution is to keep track of the incremented disparity as a floating point

Figure 4.5: Point sampling and then reprojecting the samples can lead to geometrical distortions.



value in the fragment stage. The pixel position of the point for each consecutive view would then be calculated by casting the float value into an integer position. The remaining positional and disparity inaccuracies can be reduced by supersampling, reprojecting the fragments and then applying an antialiasing filter [25]. However, since the points are rendered with gaussian splats, this is effectively applying a low pass filter, therefore only supersampling is required.

Visibility

For reasons already discussed, occlusion handling was not required for the data set we rendered. However, the algorithm can be extended to incorporate this mechanism. There are two solutions to this problem that require further investigation to determine which would be better.

The most obvious method is to pre-sort the particles so that points furthest away get

rendered first, and to implement painter's algorithm. Sorting the particles in this way can be processor intensive, and therefore the costs may only be worth it, for large numbers of views. However, some blending functions require a depth sort, so in these cases occlusion handling becomes free.

An alternative method, which does not require any sorting, is to store a z-buffer for each view and perform z-culling on the fragments. This method's effectiveness depends on the viewing resolution. For example, if a large number of high resolution views are required then the memory cost can become significantly expensive. If however, the views must share the available resolution, this method will not be more expensive than implementing a z-buffer for a standard 2D rendering pipeline.

4.4 Results and evaluation

4.4.1 Image output

Due to current hardware limitations, i.e. the lack of array indexing at the fragment stage, it is not possible to implement the algorithm using the graphics card. However, to prove the concept of incrementing the fragments to generate novel views works, we implemented a particle rendering system which calculates the perspective projection coordinates of all the points for one view, and increments these positions for the novel views before sending them to the graphics card. Figure 4.6 shows three different viewpoints rendered from an initial view by incrementing the positions of the particles. The output was identical to a conventional single view rendering pipeline, as indicated by comparing difference images.

4.4.2 Performance analysis

An analysis can be made of all the calculations required in the process of rendering the point database using a conventional 2D rendering pipeline, where none of the calculations are shared across the views, and using our incremental fragment algorithm. This should give an estimate of the potential benefits of the new algorithm.

Performance of conventional rendering pipeline

Once the data has been sent the graphics card, each point must go through the following stages: modeling transformation, trivial clipping, lighting, mapping to 3D viewport, rasterisation and then updating the frame buffer. The number of floating point operations (FLOPS) are estimated for each stage so that a comparison can be made between the incremental fragment algorithm and the more convention single view rendering method. Most of the estimations for the number of FLOPS required is taken from [47].

4.4.3 Modeling and viewing transformation

This stage involves rotating and scaling the points from local coordinates into world coordinates. Typically, all the transformations, including projection, are transformed with one matrix, which is the concatenation of the individual transformation matrices. As well as transforming the vertices, the normals associated with the points must also be transformed so they can be used correctly in the lighting stage.

Multiplying a homogeneous point by a 4 x 4 matrix requires 16 multiplications and 12 additions. After projection, calculating the point size of the splat and clamping it within the maximum and minimum pixel size requires approximately 25 FLOPS. In total for this stage, transforming a single vertex requires 53 FLOPS.

4.4.4 Trivial accept/reject classification

In this stage each primitive is tested to determine whether it lies completely within the view volume or completely outside. However, since point rendering only deals with point primitives, further clipping, which usually takes place further down the pipeline, is not required. In cases where the splat is greater than a pixel in diameter, the rasterization process will trivially avoid creating fragments outside the viewing volume (see the OpenGL specification version 2.1).

Each transformed vertex must be tested against the six bounding planes which represent the view volume. The near and far planes are trivial as they are parallel to xy plane and only need a comparison operation each. Testing a vertex against the 4 other bounding planes involves calculating the dot product of the homogeneous point with the 3D plane

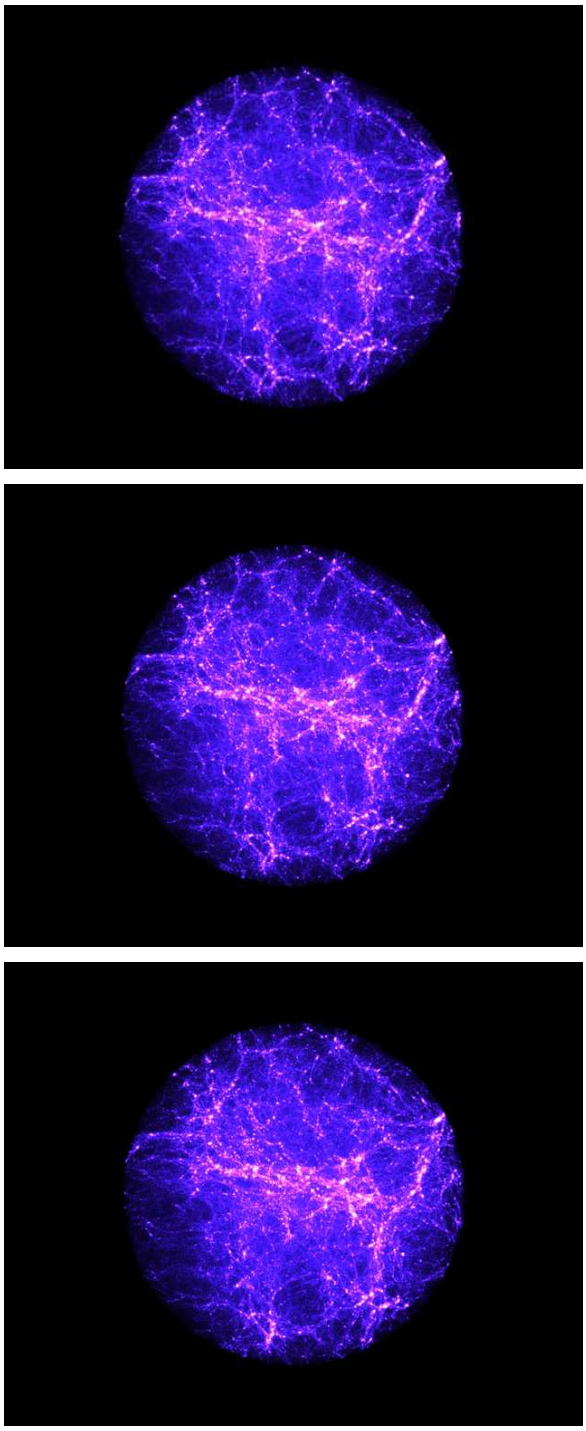


Figure 4.6: Three different viewpoints rendered from a sphere of approximately 2.6M particles cut out of the millennium simulation [145] at a redshift of 2.4.

equation, and requires 4 multiplications, 3 additions and 1 comparison operation each. In total 16 multiplications, 12 additions and 6 comparison operations are required per vertex.

4.4.5 Lighting / colouring effects

Lighting is performed on each point at the vertex stage with the mix(x,y,a) function which actually uses the following interpolation function: $x(1.0-a)+y.a$ and requires 12 FLOPS. However, the temperature of each particle must be converted into the range of [0.0, 1.0] first which requires a further 2 FLOPS.

4.4.6 Division by w and mapping to 3D viewport

After the projection transformation, the homogenous points must have each of their x, y, and z components divided by the w component. The x, y coordinates of each point can then be mapped to the coordinate system of the 3D viewport with a scaling and transformation operation. Therefore, for each vertex, 3 divisions, 2 multiplications and 2 additions are required.

4.4.7 Rasterisation and updating the frame buffer

Splatting with textured point sprites involves applying the following formulas for each fragment associated with the point sprite to determine the texture look-up coordinates, s and t (as described by the OpenGL specification version 2.1):

$$s = \frac{1}{2} + \frac{x_f + \frac{1}{2} - x_w}{size}$$

$$t = \frac{1}{2} + \frac{y_f + \frac{1}{2} - y_w}{size}$$

Where *size* is the width and height of the point sprite, x_f and y_f are the (integral) window coordinates of the fragment and x_w and y_w are the exact unrounded window coordinates of the vertex representing the point sprite. Assuming on average, the points have a diameter of 4 pixels, then each point will require 128 FLOPS. Finally blending each fragment requires an additional 128 FLOPS.

The total number of FLOPS per particle for all the stages and for n number of views is $364n$ FLOPS.

Performance of incremental fragment algorithm

The two stages, in which the incremental fragment algorithm differs from the standard rendering pipeline, for multiview rendering, are the viewing frustum set-up and the pixel shader stage.

The viewing frustum must be enlarged to encompass all the data viewable from the extreme left and right views. The amount of extra points which need rendering initially, depends entirely on the number of views and camera separation. The worst case scenario would be a 180 degrees of ”look-around” available across the views. If the viewing angle of one view is 60 degrees, then the enlarged viewing frustum may require up to three times the processing time. However, typically the viewing range between the left most and right most view is not more than 20 cm. For a typical scene with 10 cm of depth behind the display, a viewing distance of 70 cm, and assuming a one-to-one mapping between the perceived depth and virtual scene, the viewing frustum need only be enlarged by about 10 degrees. However, here we assume the entire data-set is visible in each view. Therefore, in these cases there is no performance penalty for increasing the size of the viewing frustum.

At the pixel shader stage the incremental fragment algorithm requires an extra disparity calculation (3 FLOPS) resulting in a total of 367 FLOPS per particle for the initial view; then every extra view only requires one addition per point to increment the position of the fragments; and finally blending the fragments per view is as normal (128 FLOPS). As the number of views increases, the number of FLOPS required per view decreases asymptotically towards $129n$ FLOPS.

4.4.8 Comparison between incremental fragment algorithm and conventional rendering pipeline

In all cases, the incremental fragment algorithm should perform better by at least a factor of 2.8 when rendering extra views compared to naïvely rendering the data without consideration for sharing any calculations between the views. The incremental algorithm also has another advantage in that all the views are rendered in one pass; therefore, the data does not have to be resent to the graphics card for each view.

As a case study we will analyse the theoretical performance of both rendering methods

for the X3D multiview display [127] with the entire Millennium Simulation [145], which has over 10^9 particles. Because of the size of the data set, it is not possible to store the data on the graphics card; therefore, the data has to be resent to graphics card for each pass. The X3D multiview display works well with only 10 views.

Conventional rendering requires approximately 3,670 GFLOPS for each frame. Each particle must be sent to the graphics card with the following attributes; position (3 floats), colour (32 bits or 4 bytes), density (1 float) and temperature (1 float). Therefore, it takes about 24 bytes of data per particle to be sent to graphics card for rendering. The time taken to transmit this data at a rate of 8000 MB/sec (maximum PCI-Express rate) is 28.6 sec. The NVIDIA 8800 GTX is supposedly capable of sustaining 330 GFLOPS. Therefore, rendering 10 frames may take $(11.1 + 28.6) \times 10 = 397$ sec.

The incremental fragment algorithm requires 28.6 sec to transmit the data and 3,750 GFLOPS to render the initial view which will take 11.36 sec. Each consecutive view will require 1,290 GFLOPS resulting in a total time of $11.1 + 28.6 + (3.9 \times 9) = 74.8$ sec. In this scenario the incremental algorithm is more than five times quicker at rendering the data for a 10 view multiview display. As the number of views which require rendering increase, the time taken to render Millennium Simulation with the incremental algorithm tends towards $39.7n$ sec whereas the incremental algorithm tends towards $3.9n$ sec. Therefore, the incremental fragment algorithm may be up to one order of magnitude faster than conventional rendering techniques.

4.5 Conclusion

In this chapter we described a novel multiview splatting algorithm for uncorrelated 3D point data sets. Reuse of the rendering calculations across the views are maximised by deferring the viewpoint generation until after rasterisation of an initial view. A hardware implementation is proposed by adding array indexing support at the shader level of the programmable pipeline. Further optimisations can be made if multiple fragments can be sent to the shader program at the same time (fragment clustering). Theoretical analysis shows that the incremental fragment algorithm is capable of rendering up to an order of magnitude more views in the same amount of time as a conventional rendering pipeline

for large data sets such as the Millennium Simulation, and in all cases should perform better by at least a factor of 2.8.

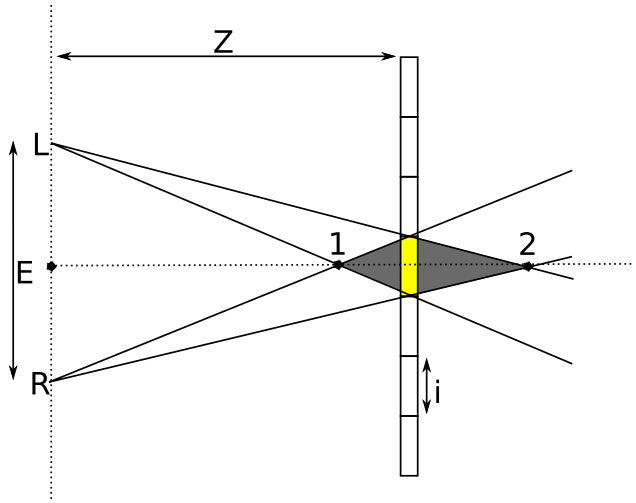
Chapter 5

Multi-layered rendering

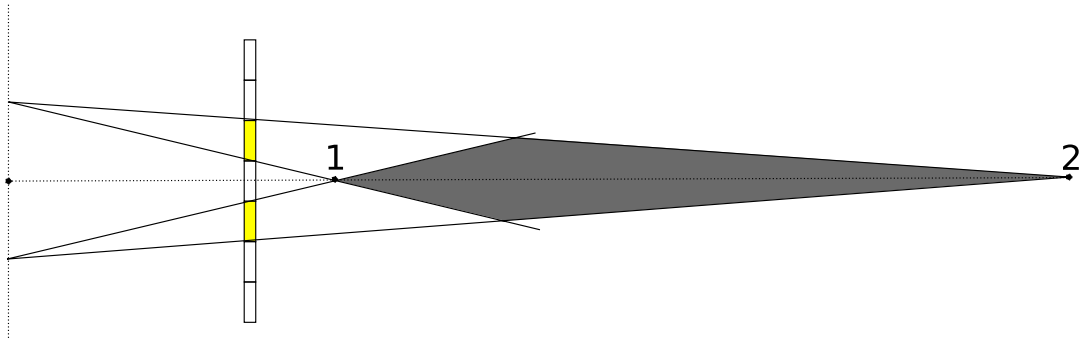
The previous chapter demonstrated how perspective coherence can be used to improve point rendering efficiencies for multiview stereoscopic images. In this chapter we describe a novel algorithm which incorporates elements from our first algorithm, i.e. exploiting perspective coherence for a point-based rendering platform, but also, uniquely, takes advantage of the fact that stereoscopic displays, especially multiview displays, tend to have a low stereoscopic resolution. The algorithm is implemented on the graphics card and works by mapping the scene volume to the desired stereoscopic resolution using multiple textures and then reprojecting and compositing (flattening) those textures to create different viewpoints; we call our algorithm the Multi-layered Renderer (MLR). The MLR algorithm offers greater rendering performance over existing solutions and also allows sophisticated control of the stereoscopic depth at little or no extra cost.

5.1 Stereoscopic / voxel resolution

We begin by defining stereoscopic resolution and then proceed to explain how what appears to be a disadvantage of stereoscopic displays can be used to our advantage by reducing the amount of computation required at the reprojection stage of the rendering pipeline. Due to the nature of stereoscopic planar displays, corresponding pixels in the left and right images are perceived as a small volume of depth, otherwise known as a voxel, (see Figure 5.1). Oddly, objects represented with zero disparity, that should in fact be flat, will in reality be perceived with a small amount of depth because the underlying, corresponding



(a) One pixel disparity



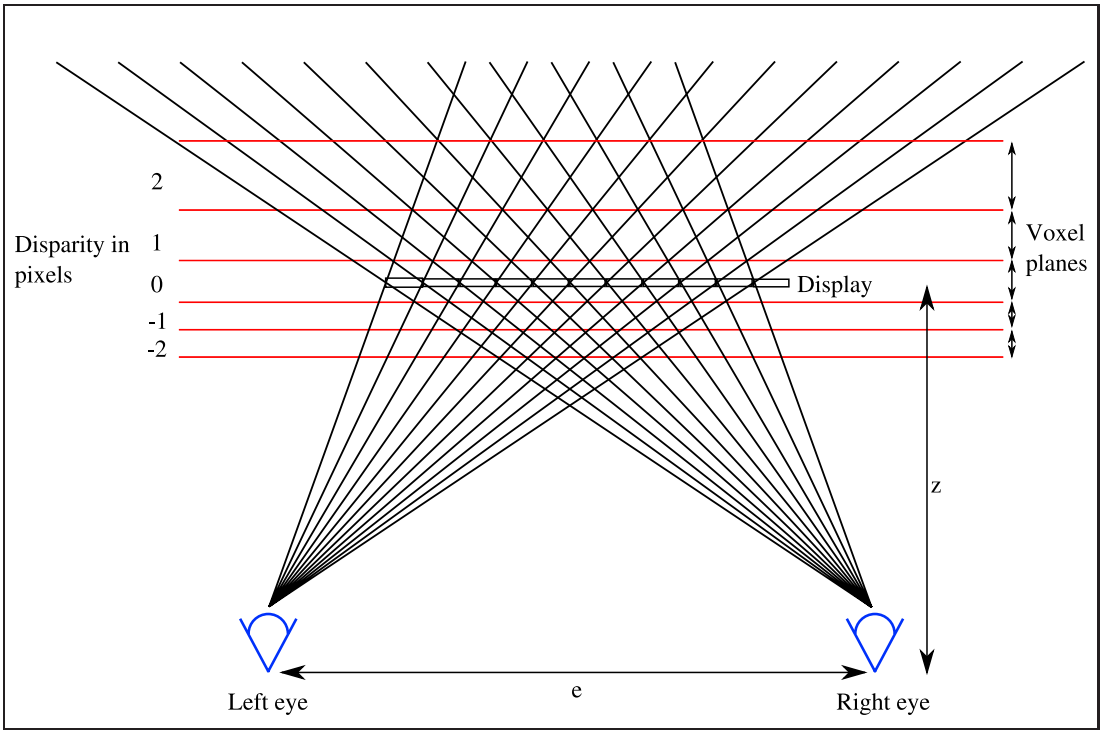
(b) Two pixels disparity

Figure 5.1: A pair of corresponding pixels in the left and right images are perceived as a volume of depth. Even zero pixel disparity will be perceived with depth because the pixels have non-zero width (see [66] for more details).

pixels have non-zero width (see [66] for more details). The result from viewing these displays is that the discrete division of the images into pixels quantizes the perceived depth into depth planes. The corresponding pixels effectively generate a three-dimensional lattice of voxels. Figure 5.2 illustrates this concept in two-dimensions. Three-dimensional displays are only capable of producing a finite number of voxels and depth planes based on the number of horizontal pixels available. The number of depth planes available within the perceived depth range is considered to be the stereoscopic resolution of the display. However, as mentioned in the background chapter, the depth range is often limited to a comfortable range, which decreases the stereoscopic resolution further.

Figure 5.2 also shows that the voxels are arranged in planes which increase in depth

Figure 5.2: Corresponding pixels viewed in stereoscopic create a three-dimensional lattice of voxels (see [139] for more details).

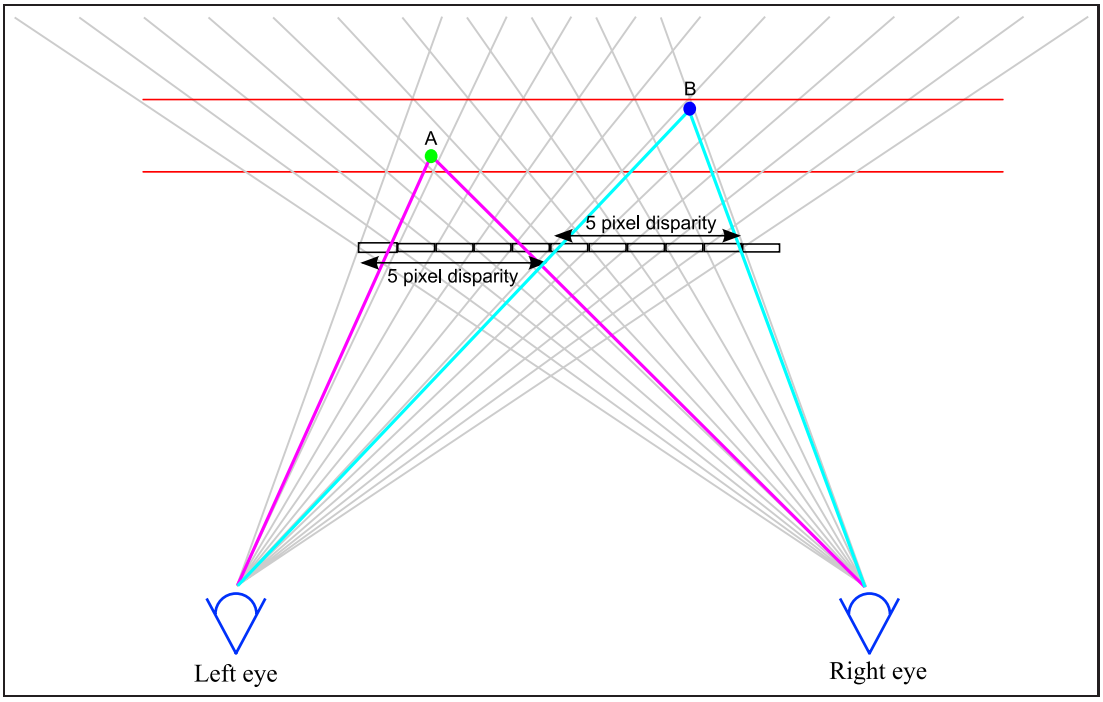


the further they are from the viewer. This fact is often overlooked but it could be critical in scientific applications such as medical systems as it means that scene depth can dictate the stereo-resolution of an object; the piece-wise linear mapping algorithms [64, 167], which can map the scene depth arbitrarily to perceived depth could potentially be adapted to take into account the stereoscopic resolution as well. However, in Chapter 6, we show how the MVR algorithm can also be used to arbitarily map the scene depth to the perceived depth range, but at no extra cost than a single region mapping approach; the algorithms [64, 167] are expensive because multiple rendering passes are required and in the worst-case as many rendering passes as depth planes are required.

The dimensions and spatial arrangements of the voxels planes are dependent on the underlying dimensions of the pixels and on the viewing properties of the observer, for example, eye separation and viewing distance of the observer. The pixel resolution per view is also a key characteristic of 3D displays as it dictates the smallest amount of stereoscopic depth that can be simulated. The perceived depth span of a voxel is the perceived

depth difference between points 1 and 2 in Figure 5.1 and can be calculated using Equations 2.1.3 and 2.1.4, described in Chapter 2. The stereoscopic resolution can be calculated by determining the screen disparity, d, required to reproduce the desired depth range, and dividing that value by the width of a single display pixel, i.

Figure 5.3: Vertex A and B have different scene depths, but because they are mapped to the same voxel plane they are projected with the same amount of pixel disparity.



5.1.1 Exploiting displays with limited stereoscopic resolution

We have shown that rendering costs for 3D displays can be reduced by exploiting the perspective coherence and only reprojecting each vertex horizontally; most stereoscopic rendering algorithms rely on this optimisation. However, notice that in Figure 5.3 all the vertices which are mapped to the same voxel plane are projected onto the display with the same amount of pixel disparity regardless of any difference in their scene depth. This observation suggests greater rendering efficiencies can be realised by grouping the vertices according to their voxel plane position and reprojecting each voxel plane instead of calculating the reprojection quantities for each vertex; we are not aware of any stereoscopic

rendering algorithms that take advantage of our observation. Depending on the scene complexity, the computation savings could potentially be much greater than algorithms that only take advantage of perspective coherence, as demonstrated in the case-study example below.

5.2 MLR design outline

We propose an algorithm that divides the scene volume up into segments according to the available depth planes, renders each segment to a texture and then reprojects those textures to generate as many new vewpoints as required; Figure 5.4 illustrates the basic steps of the MLR, which are as follows:

- **Depth mapping (scene volume division):** the first stage of the algorithm involves calculating the available number of depth planes or stereoscopic resolution of the display and then determining which voxel plane each vertex in the scene should be mapped to according to our desired depth range and a depth mapping function. The perceived depth of the scene can be manipulated arbitrarily at this stage; for example, we can represent a region of interest with greater stereoscopic resolution by mapping that region of the scene to a greater share of the available depth planes.

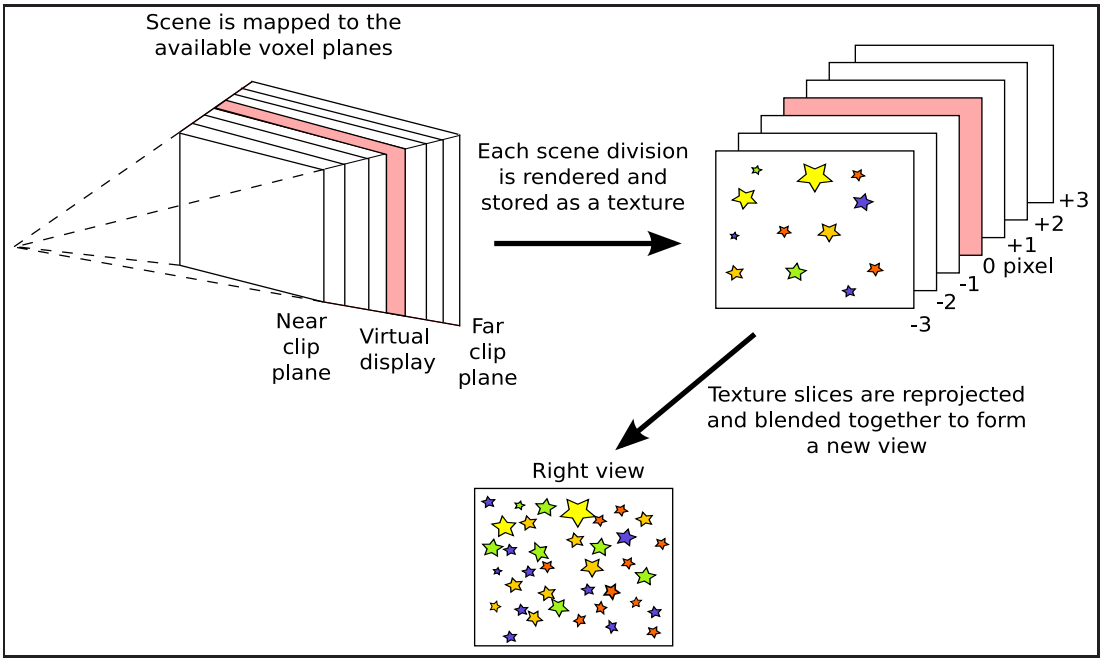
- **Rendering texture slices/depth planes:** in the second stage each previously calculated division of the scene is rendered to a different texture (called texture slices).

- **Reprojection and compositing:** the third stage is the synthesis of viewpoints by reprojecting the texture slices by different amounts of disparity and compositing (blending) them into a single image.

5.3 Stage 1 - Depth mapping (scene volume division)

In stage 1, the scene volume must be divided up into segments by using a depth mapping function, e.g. single region mapping [83]. In order to perform this division, the spatial properties of the voxel plane lattice must be known or derived and then mapped onto the

Figure 5.4: The MLR involves three stages: the scene volume is initially divided up and mapped to the available depth planes; each volume division is then rendered to a texture; finally the textures are reprojected with the appropriate amount of disparity and composited to generate the viewpoints.



scene volume. The depth of a voxel plane relative to the distance of the observer to the display screen is described in [139] with the following equation:

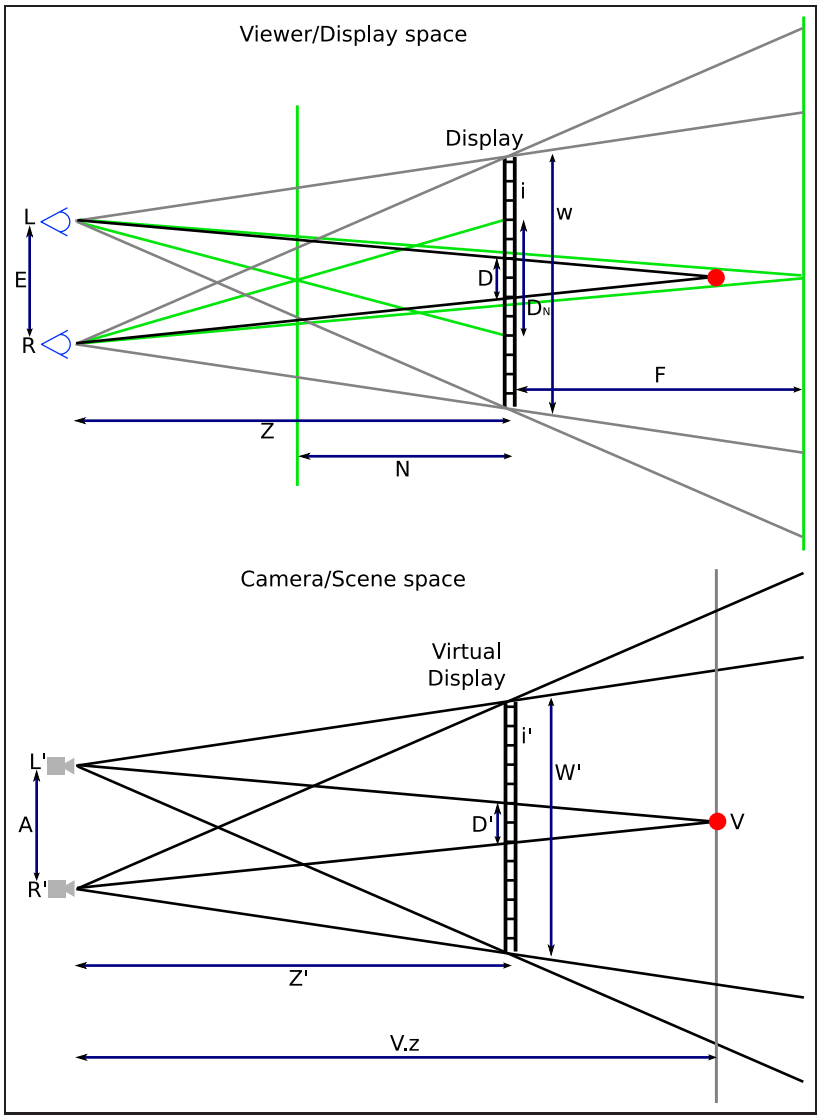
$$Z_{rel} = \frac{1}{1 - \frac{Di}{E}} \tag{5.3.1}$$

where D is the disparity at the display in pixels, i is the width of one pixel, and E is the eye separation (see Figure 5.5 for an illustration of the arrangement of the viewer, display and parameters in the equations presented in this section).

5.3.1 Adapting the single region mapping

If we wish to control the perceived depth in a similar manner to the single region mapping described in [83], a number of changes and additional steps must be made. Initially the camera separation is calculated in exactly the same manner as in [83] i.e. $A = \frac{sD_N N'}{Z' - N'}$ and $s = \frac{W'}{W}$. With the camera separation known, the screen disparity for each vertex in the

Figure 5.5: Geometry of the the viewer/display and camera/scene space: E is the eye separation, Z is the viewing distance of the observer to the display; D is the screen disparity in pixels; i is the width of a single pixel; N and F are the furthest distances each side of the display at which objects should appear to the viewer; W is the width of the display screen; w' is the width of the virtual display; A is the camera separation; Z' is the distance from the camera to the virtual display; $V.z$ is the distance of the camera to the virtual vertex V.



scene can be calculated, which ultimately determines the voxel plane the vertex resides in. Equation (5.3.1) can be rearranged to give:

$$D = \frac{E(Z_{rel} - 1)}{iZ_{rel}}$$

This allows the disparity to be calculated for any point in the viewer space. The virtual disparity of a vertex in the scene can be calculated similarly:

$$D' = \frac{A(Z'_{rel} - 1)}{i' Z'_{rel}} \tag{5.3.2}$$

where i' is the width of the virtual display divided by the horizontal pixel resolution, i.e., a virtual pixel. Z'_{rel} is the depth of the vertex, V, relative to the virtual display or zero parallax plane, and is calculated by:

$$Z'_{rel} = \frac{V.z}{Z'} \tag{5.3.3}$$

Equation (5.3.3) relies on the fact that the disparities on the display screen and virtual display are in proportion (see [83] for proof). Determining which voxel plane a vertex resides in then involves substituting the appropriate values into equation 5.3.2.

5.3.2 Calculating the number of texture slices

Rearranging the GPD model equations 2.1.3 and 2.1.4 so that the screen disparity is the subject gives:

$$D_F = \frac{P_F E}{Z + P_F} \tag{5.3.4}$$

$$D_N = \frac{P_N E}{Z - P_N} \tag{5.3.5}$$

The stereoscopic resolution can be calculated by substituting the viewing parameters into equations 5.3.4 and 5.3.5, and dividing the total disparity by the width of a single pixel on the screen.

5.4 Stage 2 - Rendering texture slices / depth planes

Each division of the scene is required to be rendered to a separate texture. In a fixed OpenGL rendering pipeline, the vertices would have to be sorted by depth and rendered in multiple passes in order to generate the texture slices. However, with programmable shaders, we can calculate the depth mapping on the fly for each vertex or fragment and decide which texture slice to update and so achieve the goal in a single rendering pass.

In order to achieve satisfactory performance, the textures must be stored in the memory of the graphics card. However, memory is limited, therefore care must be taken to

obtain the desired depth range and image quality without exceeding the available storage capacity. In cases where there is not enough texture memory, one of the following attributes must be reduced: maximum depth range; pixel resolution of each view; colour quality, i.e. number of colours or bits per channel; or the stereoscopic-resolution.

5.5 Stage 3 - Reprojection and compositing

In order to recreate a stereoscopic image, each texture slice must be reprojected by a certain amount of disparity before being composited into a single image. For the viewer to see a correct stereoscopic image, neighbouring texture slices associated with the depth planes should only be separated by a single pixel of disparity; this results in a stereoscopic image which makes use of all the available depth planes. The texture slices are initially rendered from the left-most view's perspective, which saves on reprojection calculations for one view and simplifies the implementation as only views to the right need to be generated. In order to generate the right view for a two-view display, we start with the texture slice associated with the zero parallax plane and increment each successively deeper texture slice and decrement each nearer texture slice (depth planes which come out of the screen and appear closer to the viewer) by a single pixel.

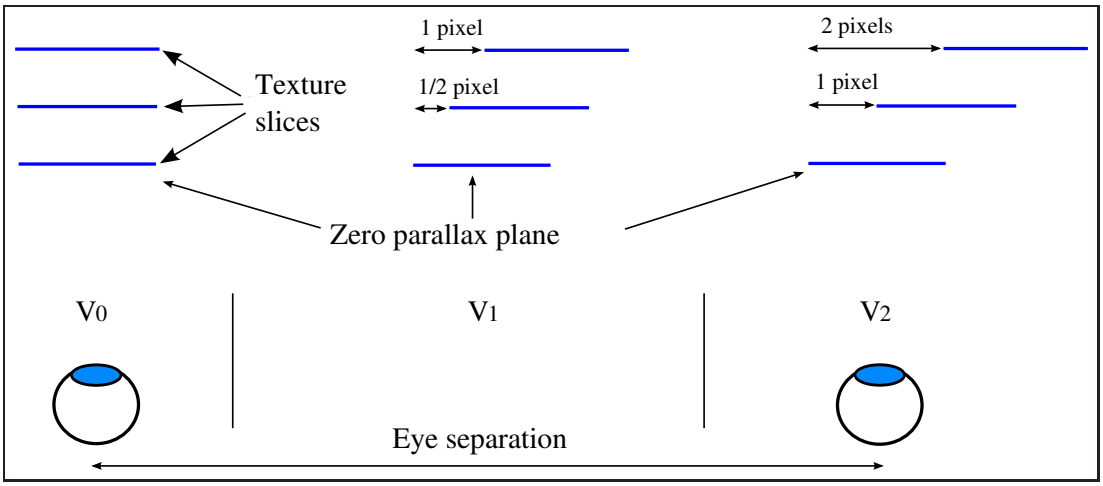
In a multiview display, at any given viewing position, the corresponding texture slices in the two observable views must be separated by the same disparity as calculated for a two view display. Therefore, the texture slices will have to be reprojected incrementally by a fraction of a pixel. In order to achieve this, the viewpoint density (which is the number of views within the interocular distance at the viewing distance) must be known or calculated first. The viewpoint density and view number then dictates the amount of disparity each texture slice is reprojected by. We use the following equation to calculate the incremental disparity for each view:

$$\frac{1}{V_p - 1} * V_n$$

where V_p is the viewpoint density, V_n is the view number, and the leftmost view is numbered zero. Figure 5.6 illustrates reprojecting three texture slices for three views.

Reprojecting by less than a single pixel is possible with texture filtering, which is a form of anti-aliasing; there are a number of methods e.g. linear, bilinear, trilinear, antis-

Figure 5.6: The amount that each texture slice is reprojected by for a multiview display depends on the view and the viewpoint density. In a two view display the depth planes are generated because the disparity is quantized into pixels; each texture slice is therefore incremented by one pixel. In a multiview display the observer should still perceive the same amount of depth regardless of the viewing position, i.e. each consecutive texture slice in the right view must still be incremented by a single pixel relative to the left view. Therefore, the texture slices must be incremented by less than one pixel for the intermediate views.



copic filtering etc, all of which have different costs and quality. Alternatively, the quality could be improved by rendering the image at a higher resolution and then downsampling it, i.e. each pixel in the final image is calculated by averaging a block of pixels from the higher resolution image. Texture filtering and anti-aliasing may improve the rendering accuracy; however, rendering results may still be different from conventional rendering techniques. This issue will be analysed and discussed in more detail in the evaluation section.

5.6 Potential performance improvements case-study

Under standard desktop viewing conditions, a twin view stereoscopic display with a resolution of 2x1280(h)x1024(v) will have approximately a stereoscopic resolution of 60 depth planes for a perceived depth range of $\pm$10 cm. Multiview displays offer even less stereoscopic resolution since often the pixel resolution is shared across the views; for

example, a single-LCD multi (9) view display with a high-definition (HD) pixel resolution of 1920(h)x1024(v) will only have a resolution of 640x360 pixels per view and approximately a stereoscopic resolution of 36 depth planes (see [66] for a more in depth discussion on the topic of stereoscopic resolution). As an example of the potential computation savings available, consider a scene containing of 500K particles; the number of reprojection calculations required for 9 images at 640x360 pixel resolution will be: 36 depth planes multiplied by the resolution per view (230,400 pixels) multiplied by 8 new views, which results in about 66M reprojection calculations. This compares favourably to our previous algorithm, considering that each particle might on average be represented with a splat of about 36 pixels (diameter of 6 pixels) and therefore would require about 144M reprojection calculations or twice as much computation.

5.7 Implementation details

The three stages, depth mapping, texture slice rendering and reprojection and compositing, were implemented using two pairs of vertex/fragment shaders in OpenGL and the GL Shading Language. The first pair of shaders are responsible for applying the depth mapping function and rendering each texture slice, while the second pair are responsible for reprojecting and compositing the texture slices. For this first implementation we only consider point rendering with Gaussian splatting and additive blending as this eliminates the requirement for occlusion handling. The code for the shaders can be found in appendix B.

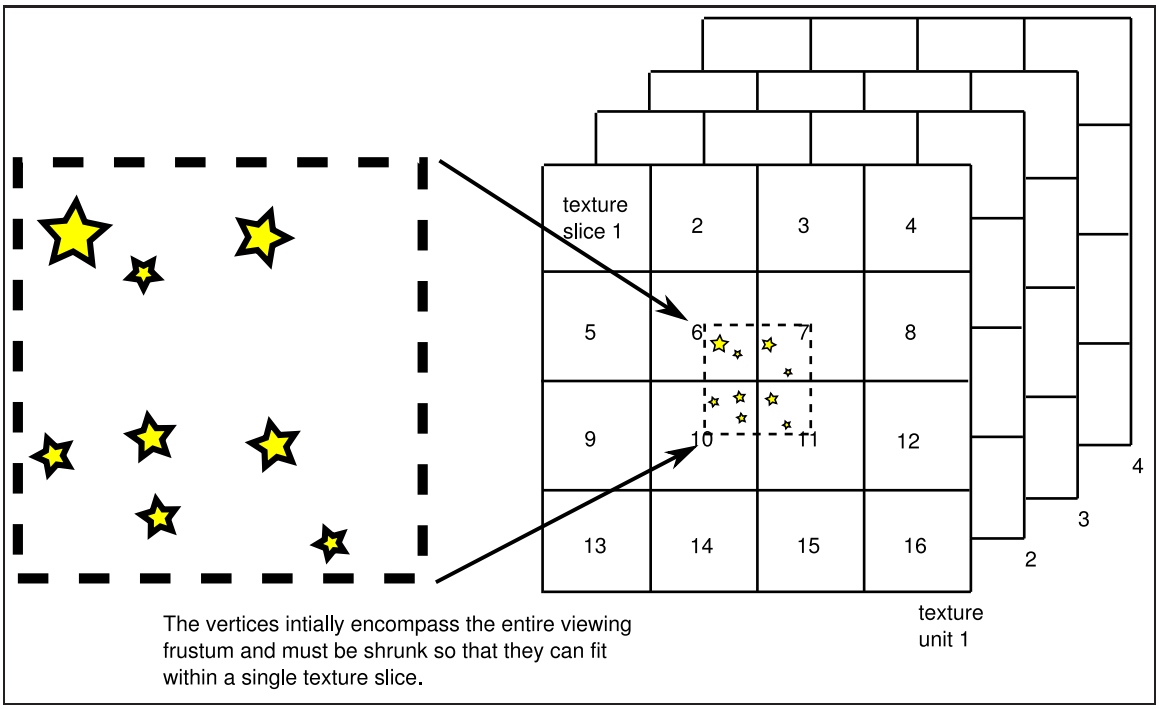
Vertex and fragment shader 1

In order to calculate the depth mapping and generate all the texture slices in a single rendering pass we decided to make use of the FrameBuffer Object extension (FBO) and Multiple Render Targets (MRT). The FBO extension is a simpler and more efficient method of rendering to texture objects, than using the pbuffer or other methods involving OpenGL context switching because it allows a number of draw buffers or textures to be attached and rendered to simultaneously. While the current OpenGL specification allows up to 16 attachment points, in practice the number is limited further depending on the hardware

and drivers of the system; it can be queried with the following snippet of code:

```
GLint maxDrawBuffers;
glGetIntegerv(GL_MAX_DRAW_BUFFERS, &maxDrawBuffers);
```

Figure 5.7: The first stage of the algorithm calculates which texture unit and also in which region of the texture a vertex should be rendered to. A number of texture units are attached to the FBO and each texture unit is large enough to fit many texture slices.



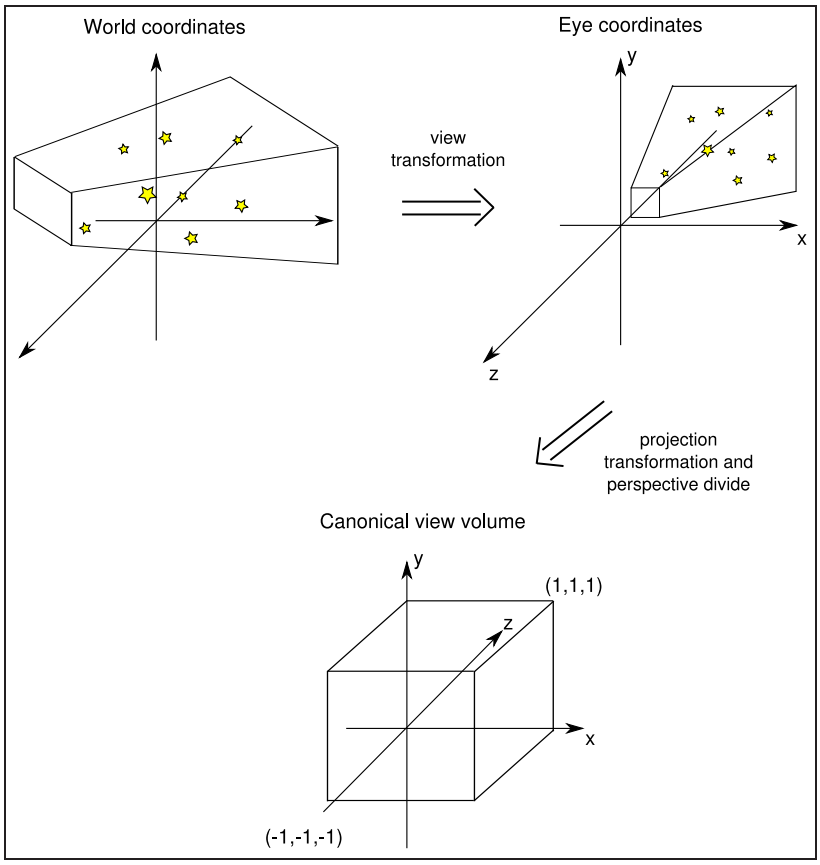
The latest graphics cards, e.g. the nvidia GeForce 8800 series and Quadro FX 5600, can only support up to 4 attachment points. This presents a problem if each slice is rendered to a different texture since realistically at least 20 voxel/depth planes are required. A naïve approach would involve using multiple FBO's, but this would increase the number of rendering passes and reduce performance.

However, an important property of FBO's is that the dimensions of the renderbuffer do not necessarily have to equal that of the viewport. Therefore, we can take advantage of this by setting up the FBO with a very large resolution and render multiple depth planes to each texture. This method requires the vertex shader to calculate which region of the texture the current vertex should be rendered to as well as which texture attachment to

use.

Figure 5.7 illustrates the concept of rendering multiple slices for each available texture unit/target with a simple example of rendering 64 slices each at a resolution of 1280x1024 pixels to 4 texture targets. Each texture target is set-up with a resolution of 5120x4096 and has space for 16 slices. Initially the scene encompasses the entire viewing frustum; therefore, the scene must be shrunk to the size of a single slice and centred to allow correct perspective projection. After the perspective projection, each vertex is transformed to the correct region in the frustum and rendered to the appropriate texture target.

Figure 5.8: OpenGL coordinate systems.



Reprojection of the vertices is delayed until after the perspective projection because otherwise each point's distance from the camera's line of sight will vary depending on which slice number it is designated. This would exhibit gross distortions in the final image since the positional difference after projection is a function of the point's distance from the line of sight and the centre of projection.

On the other hand, it is preferable to calculate which depth plane and therefore which texture unit a vertex belongs to after the model view transformation matrix has been applied but before perspective projection, i.e. in the eye coordinates. There are two motivating factors for this strategy. Firstly, perspective projection and perspective division maps the depth of the vertices in a non-linear manner which would require extra computation to reverse, in order for the equations described above to still apply. Secondly, calculations can be simplified if a fixed coordinate system is assumed, i.e. the camera is based at the origin looking down the negative z-axis. Figure 5.8 illustrates the coordinate systems used in OpenGL's rendering pipeline.

The fragment shader now has to write the fragments to the correct texture buffer/unit. The vertex shader passes on this information by using a varying float to dictate which texture unit the vertex's fragments should be rendered to. However, there are two issues which must be dealt with. Firstly, sending data using the varying attribute may lead to some imprecision in the value passed. To compensate, a small range check is made to determine which texture unit value is being passed to the fragment shader. Secondly, fragments must be rendered to every attatched render target or discarded completely; a fragment can not be rendered to the required texture unit and discarded for the rest. We solved this problem by assigning zero opacity to each fragment that needs discarding (i.e. the fragment's colour will not contribute to the pixel in the blending stage).

Vertex and fragment shader 2

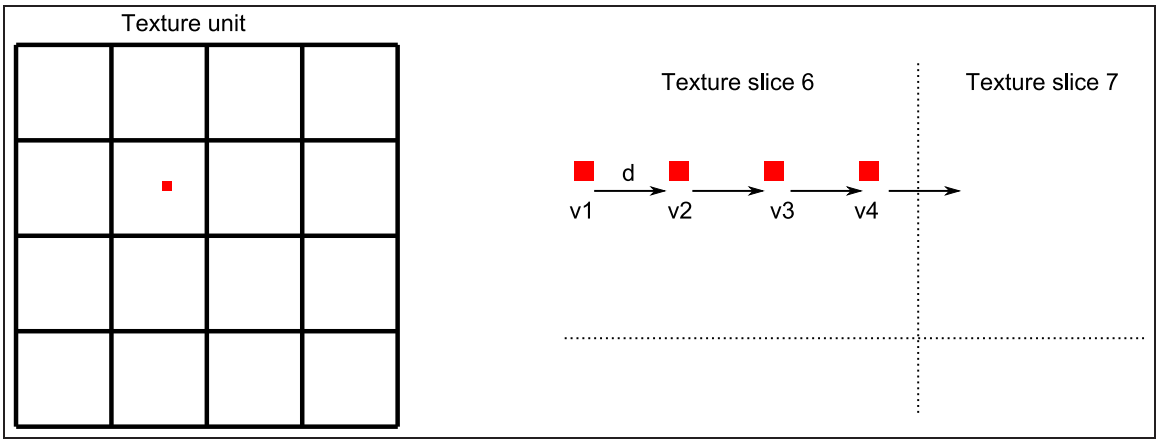
The final reprojection and compositing stage involves generating multiple views by applying multi-texturing to a single quad designed to encompass the entire viewport. For each view, the fragment shader is responsible for calculating the correct texel (a single pixel from a texture) coordinates from all the texture slices and to blend them into a final image of the same resolution as the viewport. The saved textures will be much larger than the desired viewport as they contain many texture slices. Therefore, the texture coordinates for the quad are assigned to only a small corner of the the texture map.

In the main program the disparity offset between each consecutive texture slice is calculated by multiplying the view number (which view is to be rendered) by the width of each texel. Texture coordinates have values ranging from 0.0 to 1.0. Therefore, the width

of a single texel is the reciprocal of the resolution of the texture unit. The view number also ranges in value from 0.0 to 1.0. The fragment shader increments the disparity offset for each texture slice by multiplying the value with the texture slice number and applies this offset to the texture coordinate before acquiring the texel. The left most view is rendered by stacking the texture slices without disparity shifting and is represented by the value 0.0, and a value of 1.0 will obtain the right most view, i.e. each texture slice is shifted by the appropriate disparity multiplied by 1.0. View values between 0.0 and 1.0 will result in intermediate views being generated.

5.8 Implementation issues

Figure 5.9: In the second stage of the algorithm each new view is generated by adding the appropriate disparity amount to the texture lookup coordinates. In some cases this may result in an incorrect texel being retrieved unless clipping is performed.



An important stage which we chose not to implement is texel clipping in the second stage of rendering. As Figure 5.9 illustrates, the fragment shader retrieves a texel for each view by adding the appropriate disparity to the texture coordinates. This can be problematic for fragments close to the border of a texture slice since incorrect texels from adjacent texture slices may be retrieved. The solution would involve boundary checks before blending each texel. However, by carefully controlling the camera placement so that the entire volume of data is visible means clipping is not always necessary; in our

case it was desirable to view the entire scene, and for performance gains texel clipping was not implemented.

Most displays are only capable of displaying up to 16 million colours. Therefore, often programs only output pixels in 24 or 32 bit format. Pixels are composed of 3 or 4 channels; red, green, blue and alpha. Each component can have a value from 0 to 255. However, blending millions of particles with such a limited range can lead to poor quality images. Frame buffer objects have an advantage in that textures can be attached with 16 bits or even 32 bits per channel. We chose to assign 16 bits per channel which allows each channel to be represented by floating point values instead of an integer and gives a much greater range but also is not too detrimental to the rendering performance. Blending with a larger range of values means less colour banding, greater detail and generally better looking images. Rendering in this manner is called High Dynamic Range (HDR) rendering (see [126] for more details on HDR imaging techniques).

Unfortunately, the pixel values must be converted back into the usual [0,255] range suitable for the display once the blending stage is over. Simply clamping values over 255 defeats the purpose of using a HDR to improve quality. Therefore, tone mapping techniques must be used to convert the HDR into a lower one. HDR and tone mapping is a large and active area of research and beyond the scope of this thesis. We applied a basic tone mapping operator in the fragment shader:

$$L = \frac{Y}{Y+1} \tag{5.8.1}$$

where L is the output luminosity of the fragment and Y is the input luminosity. The function maps values from the range of [0,∞] into [0,1].

5.9 Evaluation

We evaluate the performance of the MLR algorithm described above by comparing the rendering times and accuracy to a naïve single viewpoint renderer (SVR) under a range of different scenarios. The SVR employs the same visual effects as in the MLR implementation but does not take advantage of any stereo-coherence available across the multiple viewpoints, i.e. each view is rendered independently of the other. Each particle was rendered using an OpenGL anti-aliased point so as to approximate splatting, similar to the

technique described by Hopf et al. [68, 69]. The same point data set described in Chapter 4 was also used in this evaluation.

5.9.1 Rendering speed

In this section the rendering times for both the SVR and MLR are compared with varying sample rates from the data set. During this stage the V-sync option controlled by the NVidia drivers was switched off. V-sync is usually desirable because it synchronizes the frame buffer updates with the vertical blanking interval of the display so as to avoid visual artifacts such as shearing and tearing. However, this option when switched on would also interfere with the rendering times and compromise the validity of any comparison between the SVR and MLR. Also linear texture filtering was used.

Figure 5.10: A comparison between the MLR and SVR algorithms for a typical 9 view multiview display. The splat size ranges from 2-6 pixels, resolution per view is 640x360 and the depth is about ± 10 cm which requires 36 texture slices for the MLR algorithm. The results show the timings for 100 rendering repetitions with varying sample rates of the data set.

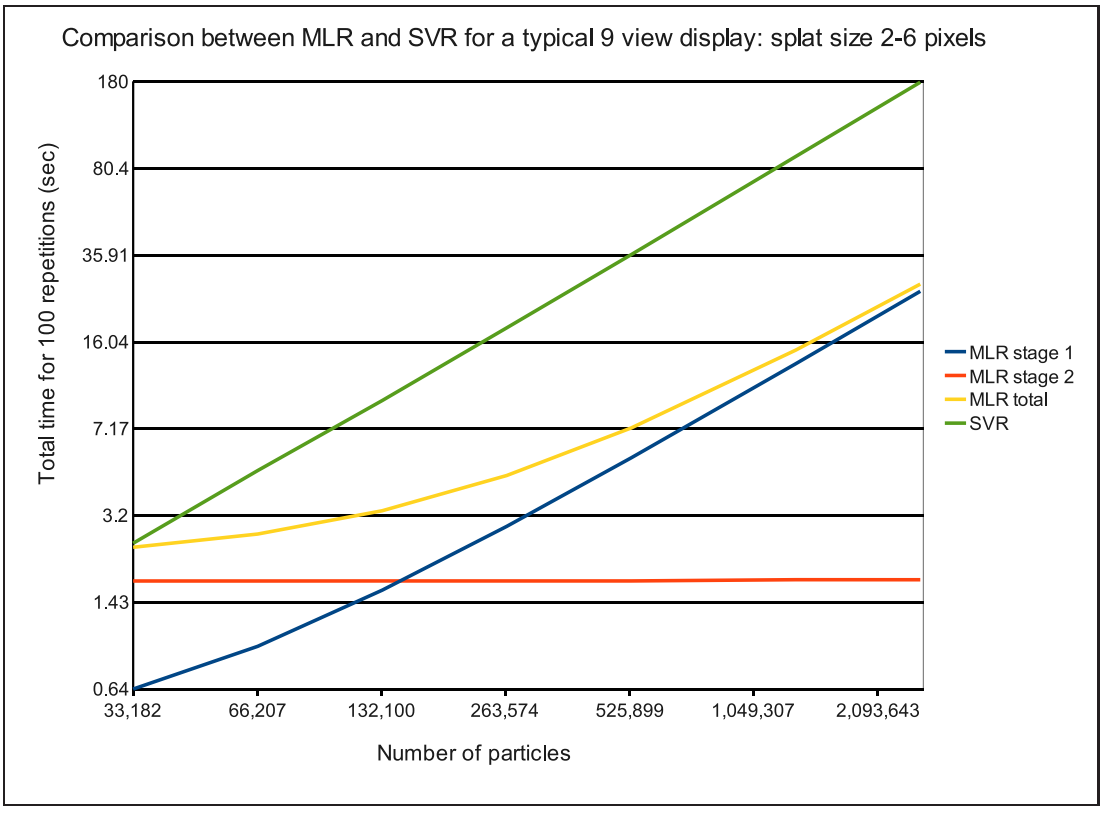


Table 5.1: Table showing the workload increase, of stage 1 of the MLR compared to rendering one view with the SVR with a splat size of 2-6 pixels.

Number of particles	Total time 100 repitions (sec)		Time increase (%)
	SVR: one view	MLR: stage 1	
33,182	0.28	0.64	133
66,364	0.54	0.95	76
132,728	1.04	1.61	55
265,456	2.03	2.9	43
530,912	4.01	5.48	37
1,327,281	9.97	13.12	32
2,654,562	19.85	25.69	29

For the first test, the paramaters were set-up for a typical HD (1920x1080 pixel resolution) 9-view multiview display. The resolution per view however, is only 640x360 pixels because most multiview displays share the pixels amongst the views from a single LCD. Also, the depth was limited to approximately ± 10 cm under standard desktop viewing conditions, which gives a stereoscopic resolution of 36 voxels.

Figure 5.10 illustrates the rendering times for the MLR (including timings for stages 1 and 2) and SVR algorithms with each particle splat size ranging in diameter from 2-6 pixels. As a reminder, stage 1 is the process of rendering the texture slices and includes the time taken to perform the first pair of vertex and fragment shaders; whereas, stage 2 is the process of reprojecting and compositing the texture slices to generate new viewpoints using the second pair of vertex and fragment shaders. We can see in Figure 5.10 that the MLR algorithm performs much better than the SVR for a large number of particles: the total time taken to render 9 views from 2.6M particles with the MLR was only 0.27 seconds compared to 1.79 seconds for the SVR which is six and a half times quicker. However, in this particular scenario the MLR algorithm is only more efficient than the SVR when rendering more than approximately 32,000 particles. The sampling rate to obtain 32,000 particles was 1.2% of the original data-set which arguably is relatively small compared to the size of an average scientific point dataset. Therefore, in the case of

displaying 3D content for a HD multiview display with a single LCD, we judge the MLR to be usefully faster than the SVR.

Table 5.2: Rendering cost of the MLR stage 2 per view for a varying number of texture slices and the average cost per texture slice. The splat size ranges from 2-6 pixels and the resolution per view is 640x360 pixels.

No of texture slices	Stage 2 (sec)	Cost per texture slice
4	0.0007	0.000175
16	0.0012	0.000075
36	0.002	0.000056
64	0.0034	0.000053
100	0.0053	0.000053
144	0.0076	0.000053

The SVR algorithm exhibits linear performance with the number of rendering primitives whereas the performance gains for the MLR initially grow as the number of particles increase and then gradually converges to a linear relationship with the number of particles. This is mainly because reprojecting and compositing 36 textures slices per view during stage 2 of the MLR has a constant cost and therefore as the scene complexity increases stage 2 becomes relatively less expensive: on average the cost of reprojecting and compositing the textures slices to generate one view is approximately 0.002 seconds.

The cost of stage 1 compared to the total cost of the SVR also becomes relatively less as the number of particles increase (see Table 5.1). We believe this is because the initial costs associated with setting up the FBO are more apparent for smaller data-sets. In this particular scenario the extra workload of stage 1 of the MVR compared to rendering a single view with the SVR appears to converge to just less than a third for large data-sets.

On our particular hardware set-up, for a typical 9-view HD display, with a stereoscopic resolution of 36 voxels, the MLR will outperform the SVR when the data set consists of at least 32K particles or the cost of rendering one view with the SVR is greater or equal

to 0.003 seconds. The performance of the MLR tends towards:

$$1.3T + 9C \tag{5.9.1}$$

where T is the cost to render a single view using the SVR and C a constant cost of 0.002 seconds for the reprojection and compositing. We can also argue that the performance loss below 32K is not important because the total time to render all the views below 32K will not take longer than 0.027 seconds which is less than 0.033 seconds, the time required for 30 frames per second and smooth animation.

Table 5.2 presents the total cost of stage 2 of the MLR with varying numbers of texture slices along with the average cost per texture slice. The resolution and splat sizes were the same as the previous example but the number of particles were kept constant at 2.6M. We can see that the cost of reprojecting and blending one texture slice rapidly converges to 0.000053 seconds. With this information in mind we can take our cost analysis one step further to a two-view HD display with a stereoscopic resolution of 128 voxels and 1920x1080 pixels per view. The cost increase of reprojecting and compositing 1920x1080 pixel texture slices compared to 640x360 pixel texture slices is about nine times more. Therefore, stage 2 of the MLR will take approximately 0.061 seconds per view ($0.000053 \times 128 \times 9$); substituting this value into equation 5.9.1 and calcuating T, tells us that the MLR will only be more efficient than the SVR when it takes at least 0.174 seconds to render a single view using the SVR ($\frac{0.061 \times 2}{0.7}$). For this situation to occur, we would need to render at least 2.3M particles ($\frac{0.174 \times 9 \times 2.6 \times 10^6}{1.79}$).

If there existed a nine-view HD display with 1920x1080 pixels per view, the MLR would be more efficient than the SVR when rendering at least 1M particles. Clearly the MLR is disadvantaged when the resolution per view is high relative to the number of particles in the data set. However, anecdotal evidence suggests a trend that GPU power increases far more rapidly than display resolution, therefore, the advantage of the MLR over the SVR for high definition stereoscopic imaging may increase in the future.

We repeated the first test but increased the size of the splats to 8-24 pixels. This mostly saturates the screen as the splats are too large, but it transfers the bottleneck in stage 1 of the MLR from the vertex shader to the fragment shader and highlights an interesting problem. Figure 5.11 reveals that while the performance of the MLR algorithm is still much better than the SVR algorithm, the performance gains are slightly less and we don't

see the break-even cost of both algorithms until approximately 34,000 particles. We can see the main increase in rendering time is due to stage 1; the time increase for stage 2 was only about one second. Table 5.3 shows that workload increase of stage 1 of the MLR compared to the cost of rendering one view with the SVR can almost be up to 300% as much. We believe the cause of the slowdown is mainly due to the fragment discarding problem, i.e. a fragment must be sent to all four render targets regardless of whether it will affect the appearance.

5.9.2 Rendering accuracy

Figure 5.12 shows the rendering output of the leftmost, rightmost and one intermediate view from the MLR and SVR algorithms along with the difference images between the corresponding views. Linear texture filtering was used. We originally hypothesised that the MLR algorithm would give identical rendering results to the SVR for the leftmost

Figure 5.11: A repetition of the first test but with the size of each splat ranging from 8-24 pixels.

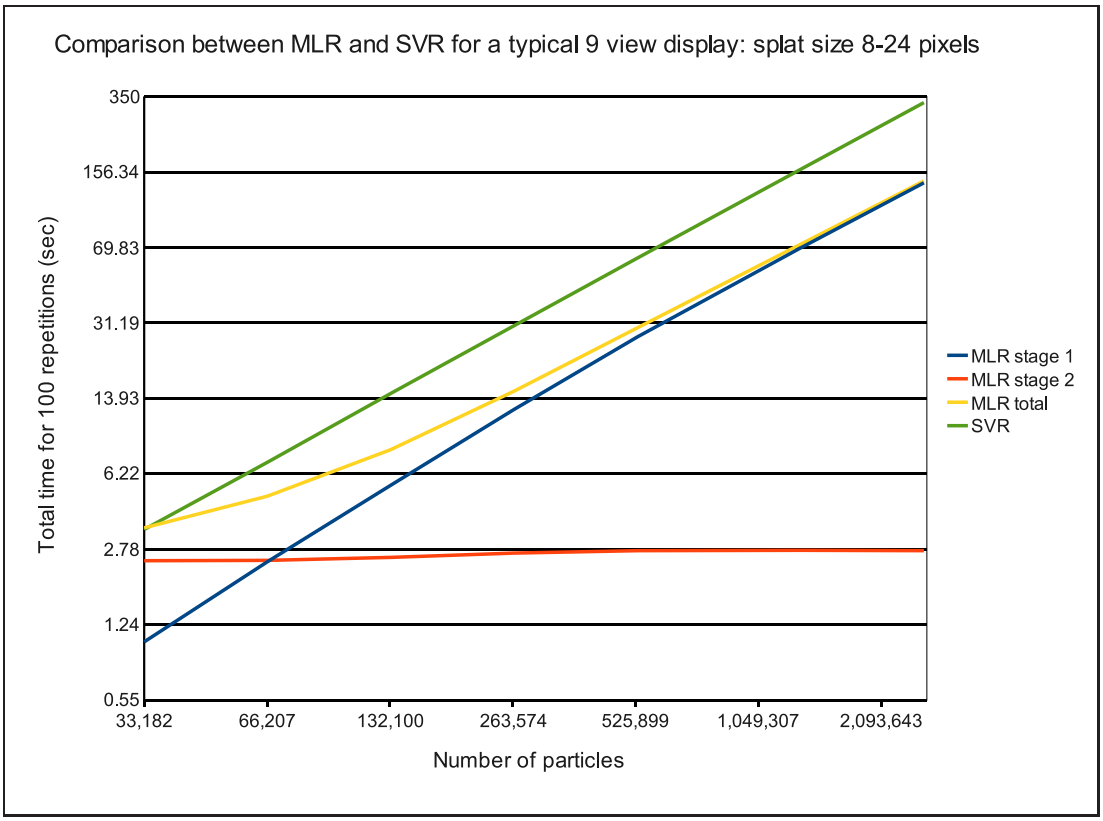


Table 5.3: Table showing the workload increase, of stage 1 of the MLR compared to rendering one view with the SVR with a splat size of 8-24 pixels.

Number of particles	Total time 100 repetitions (sec)		Time increase (%)
	SVR: one view	MLR: stage 1	
33,182	0.38	1.03	171
66,364	0.78	2.44	211
132,728	1.63	5.51	238
265,456	3.37	12.39	268
530,912	6.93	26.88	288
1,327,281	17.87	69.49	289
2,654,562	36.47	139.31	282

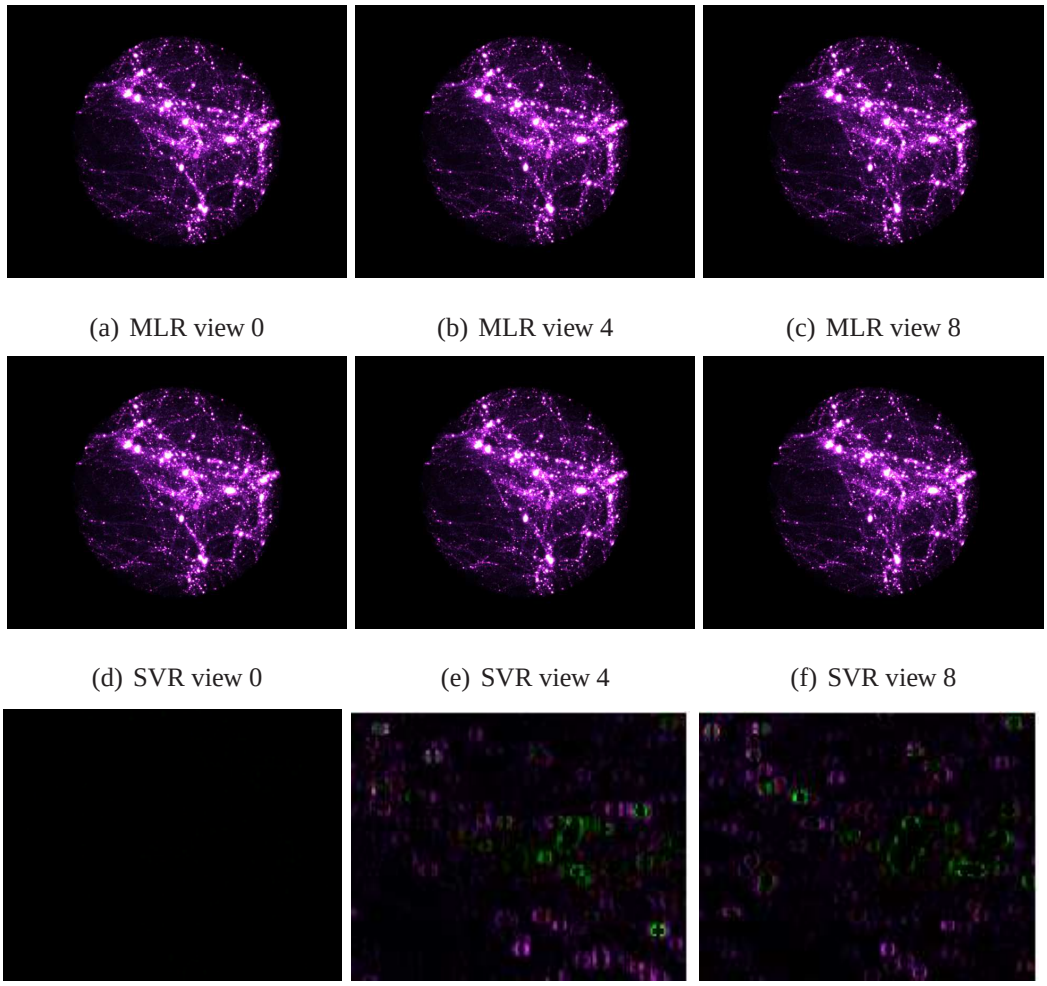
and rightmost views but that some error might occur for the in-between views due to sub-sampling the textures. The results confirmed our hypothesis for the left-most and intermediate views, but surprisingly there were visual errors for the right-most view.

Since there are no visual differences between the MLR and SVR algorithms for the left-most view we can assume the blending implementations are effectively identical. Therefore, the position errors are most likely due to either incorrectly calculating which texture slice each each vertex belongs to, or the texel fetching position for each fragment.

In order to further investigate the unexpected inaccuracies, we disabled all visual effects e.g. blending, anti-aliasing, etc, fixed the splat size to a constant value and compared the MLR and SVR rendering results from a much smaller test data set. Also the points were coloured differently by each algorithm so that the difference images could give more information on the direction of the positional errors. A close-up of the results can be seen in Figure 5.13.

Figure 5.13(d) reveals that the errors, in the right-most view, are caused by positional differences of up to one pixel in either horizontal direction. The splat sizes in the right-most view do not vary, which would be the case if the texels' positions were calculated incorrectly. Therefore, we suspect the positional errors are caused by occasionally incorrectly determining which texture slice a vertex belongs to probably because of floating

Figure 5.12: Rendering results from the MLR and SVR algorithms showing the left-most, rightmost and one intermediate view with the data set described above. Difference images between the MLR and SVR outputs are also shown.



(a) MLR view 0 (b) MLR view 4 (c) MLR view 8

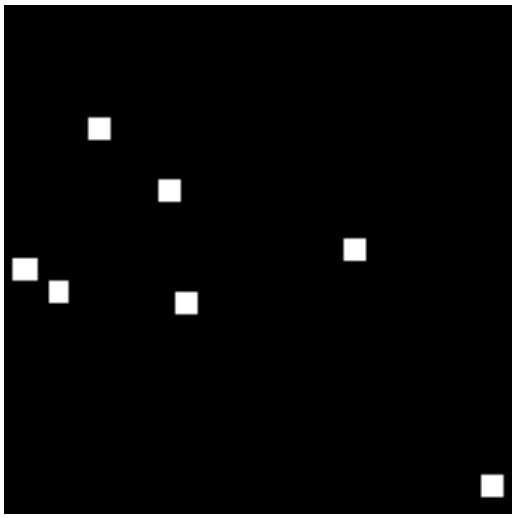
(d) SVR view 0 (e) SVR view 4 (f) SVR view 8

(g) Zoomed in difference image: view 0 (h) Zoomed in difference image: view 4 (i) Zoomed in difference image: view 8

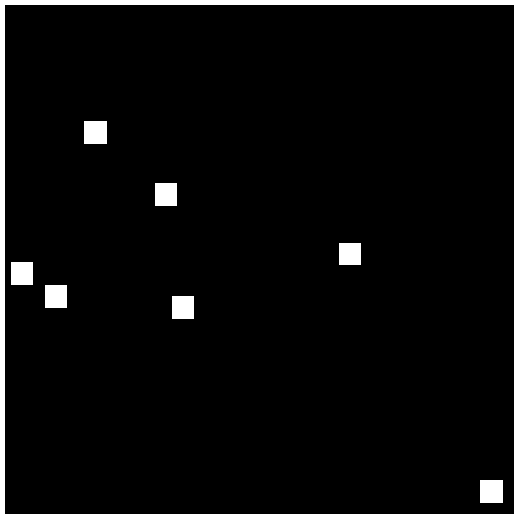
point imprecision and rounding errors.

In Figure 5.13(c) we can see that the same area of the scene contains a greater percentage of errors in the intermediate view than the rightmost view. Upon close examination of the intermediate view, we also discovered that the size of the splats are not consistent and can grow or shrink by up to one pixel. Although errors of only up to one pixel were observed, theoretically the maximum disparity error could be up to two pixels because disparity errors can occur from two sources: incorrect texture slice placement and intermediate view sub-sampling errors. However, perceived depth distortions should be

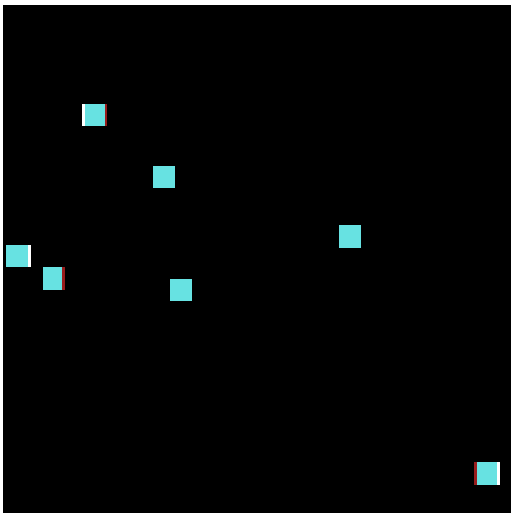
Figure 5.13: Positional differences with blending and anti-aliasing disabled on a simple test data set.



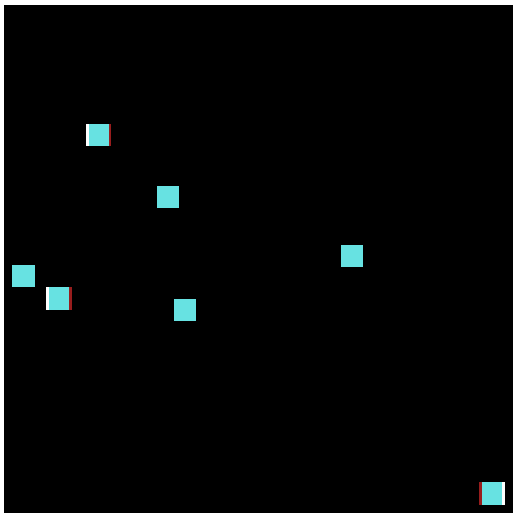
(a) MLR view 4



(b) MLR view 8



(c) Difference image between MLR and SVR: view 4



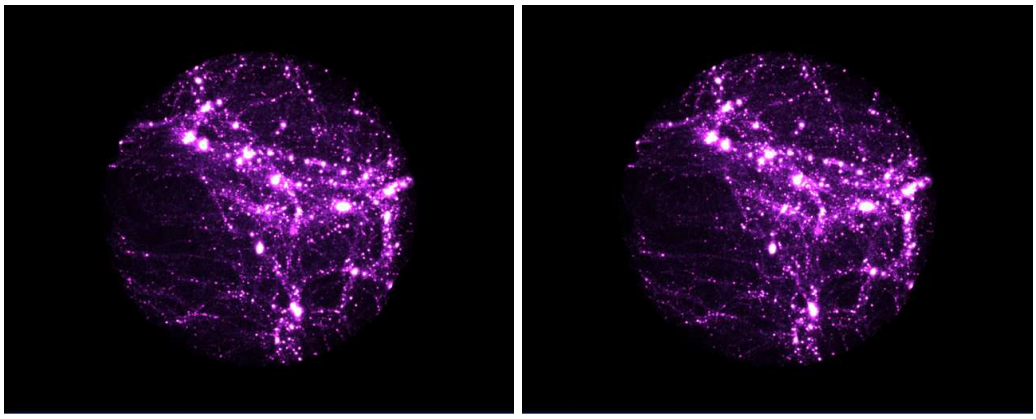
(d) Difference image between MLR and SVR: view 8

lessened when blending is enabled since the Guassian splatting technique is effectively a from of anti-aliasing.

Texture filtering

We have noted that the intermediate views between the left-most and right-most views appear to exhibit greater inaccuracies probably due to sub-pixel sampling and that some

texture filtering techniques could potentially improve the rendering results by effectively anti-aliasing the texels. We explore the costs and benefits of following common texture-filtering techniques in relation to the MLR: linear texture filtering, which is also known as nearest-neighbour interpolation; bilinear filtering; trilinear filtering; and anisotropic filtering.



(e) MLR bilinear filtered view 4

(f) MLR bilinear filtered view 8

(g) Close-up difference image between bilinear filtered MLR and SVR: view 4

(h) Close-up difference image between bilinear filtered MLR and SVR: view 8

Figure 5.14: Rendering results with bilinear texture filtering.

Linear texture filtering works by assigning the closest texel to the pixel centre. This was the method originally implemented in the MLR algorithm because it is relatively fast compared to other texture filtering methods. Bilinear filtering is probably the most basic method of anti-aliasing. Each pixel is coloured by averaging the four nearest texels to the pixel centre in a weighted average fashion according to the distance. Trilinear filtering is primarily used to alleiviate visual artifacts noticeable with bilinear filtering

when the renderer switches from one mipmap to another. However, the MLR does not use mipmapping; therefore, trilinear filtering will be of no greater benefit than bilinear filtering. Anisotropic filtering further improves visual appearance over trilinear filtering when viewing the texture at an angle because it calculates the correct trapezoid shape of the texel (bilinear and trilinear filtering always assume a square texel). In the MLR algorithm there would not be any benefit with this method over bilinear filtering as the textures are only ever viewed head-on.

We only implemented bilinear filtering because the other methods described do not offer any further benefit to the MLR. Figure 5.14 illustrates the right-most view and an intermediate view along with the difference images associated with SVR. Visually the quality of the images appear to have improved slightly; however, it is difficult to verify solely using the eye. Therefore we applied a statistical analysis tool, Perceptual Image Diff, on the images to count the number of pixels that differ (details of the program can be found at http://pdiff.sourceforge.net/).

Table 5.4: Accuracy comparison between bilinear and linear texture filtering.

View	Number of different pixels	
	Linear filtering	Bilinear filtering
0	438	2984
1	25962	8442
2	27844	9676
3	25057	11268
4	30786	13342
5	26279	15763
6	29253	18236
7	29261	20640
8	19193	21744

Table 5.4 shows the inaccuracy count from the Perceptual Image Diff tool using bilinear and linear filtering. As expected, with the linear texture filtering the errors increase the most towards the middle intermediate view and decrease slightly for the right-most view.

With bilinear texture filtering the inaccuracies steadily increase towards the right-most view and also show quite a few inaccuracies for the left-most view. It should be noted that the pixel difference count is a some what of a flawed method of analysis for comparing the anti-aliased results because some of the differences are probably desirable (i.e. smoothly blended pixels). However, on average the bilinear filtering resulted in less measurable inaccuracies than with linear filtering, therefore we can conclude with a reasonable degree of certainty that bilinear texture improves visual quality. Unfortunately the speed cost for bilinear filtering was 0.5608 seconds per view compared with 0.2 seconds per view for linear filtering.

5.10 Conclusions

In this chapter we have presented a novel algorithm that reduces the number of reprojection calculations for large point data sets compared to traditional stereoscopic rendering algorithms by grouping the vertices into their respective depth planes and calculating the reprojection quantities once per voxel plane instead of for each vertex. While the rendering performance of the MLR varies with the number of depth plains, display resolution, and number of particles, for a typical 9-view multiview display the rendering time for 2.6M particles using the MLR was shown to be more than six times quicker than the SVR; the optimum scenario for the MLR is low resolution multiview displays. A significant benefit of the MLR includes flexible multi-region depth mapping at little or no extra cost, whereas achieving this with traditional rendering methods would involve setting up multiple cameras and performing multiple rendering passes at considerable cost. A disadvantage of the MLR is that it introduces positional rendering inaccuracies; however, each rendered point can only ever be positioned incorrectly by a maximum of two pixels, and this effect can be ameliorated by using Gaussian splatting. Further potential benefits of the MLR are presented in the next chapter.

Chapter 6

Further applications of the MLR algorithm

In this chapter, we discuss further, the potential applications and benefits of using the MLR algorithm. We also make an initial attempt at describing how to implement occlusion handling and further performance optimisations available.

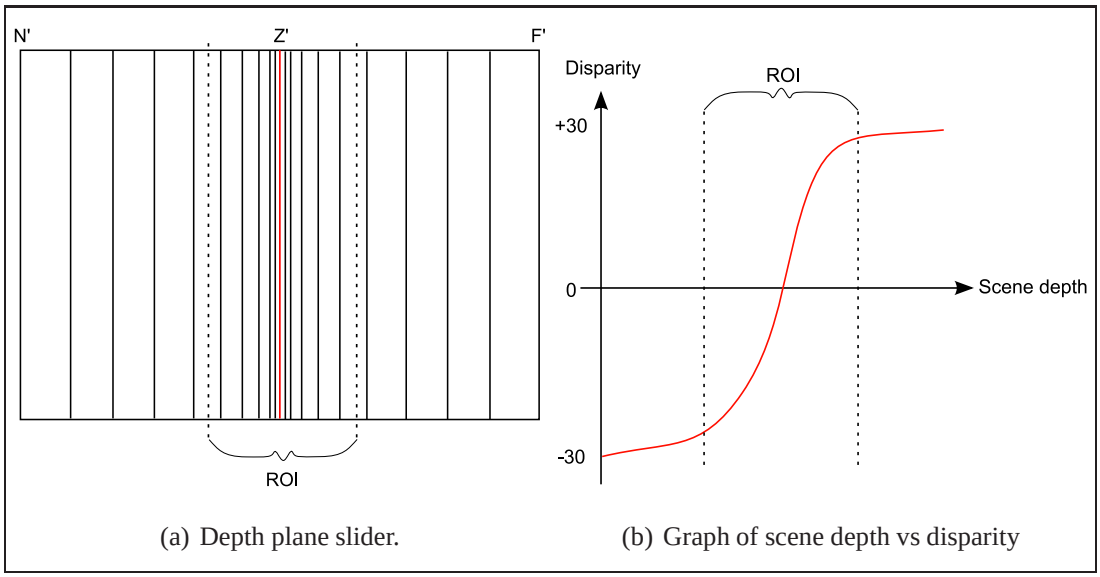
6.1 Multiple region depth mapping

As mentioned in the previous chapter the MLR algorithm is capable of providing an arbitrary number of regions in which the perceived depth can be mapped to. This technique, known as multiple region depth mapping (also described in [64, 167]), can be used to assign regions of interest greater amounts of stereoscopic resolution and is especially advantageous when either a small depth range is desirable or the display is of low resolution. Known multi-region depth mapping algorithms require each region (defined by having a different camera separation) to be rendered as a completely separate pass and the results merged; this is potentially very expensive if many regions are desired. However, because the MLR algorithm works by reprojecting each voxel plane by an arbitrary amount of disparity, multiple region depth mapping comes free; the cost is the same whether we use a single region or a multiple region mapping approach.

We believe the simplest method of controlling the perceived depth effectively, when an arbitrary number of regions of interest are allowed, is to provide the user with a graphical

interface. This interface could show the available voxel planes mapped onto the scene and allow the user to adjust the volume each voxel plane encompasses. Alternatively a graph can be drawn to show how much depth each voxel plane represents and allow the user to manipulate the graph to create regions of interest represented by more depth planes. The function of the graph would be used by the algorithm to calculate which depth plane a vector or particle belongs to. Figure 6.1 illustrates these two methods. Obviously the exact method of implementing the multi-region depth control interface is open to further investigation.

Figure 6.1: Two possible interfaces for controlling the depth using multiple regions to create regions of interest.



(a) Depth plane slider. (b) Graph of scene depth vs disparity

6.2 3DTV with Custom depth control

Synthesizing/rendering 3D content on the fly using a PC has the significant advantage of allowing the user to adjust the perceived depth range with relative ease. This is important because peoples' eye separations and depth range tolerances vary widely [35]. However, currently content for 3DTV is broadcast as a stereoscopic pair, requiring polarizing glasses to see the depth effect. The content is pre-rendered or captured and therefore, disparity is scaled by the size of the TV. Without the ability for the observer to adjust the

disparity, content creators must take into careful consideration: display size ranges; eye separations ranges; and viewing distance ranges. The simplest option to ensure acceptable stereo quality for the majority of the public is to severely limit the amount of the disparity: this may actually be a sensible approach because stereo depths as little as 2 cm can still have a significant impact (see Chapter 3).

There are a number of strategies available to ameliorate the disparity scaling issue. For example, varying perceived depth ranges amongst viewers due to different eye separations can be reduced by using high density multiview displays (see Section 2.2), however this is an expensive solution. Alternatively the disparity can be reduced by a number of post image filtering techniques, for example [90]: This method involves identifying the corresponding pixels in the 3D image pair and uses image processing techniques to interpolate the pixels to create new views based on the user's depth range preference. However, as with most stereoscopic interpolation techniques, quality is poor.

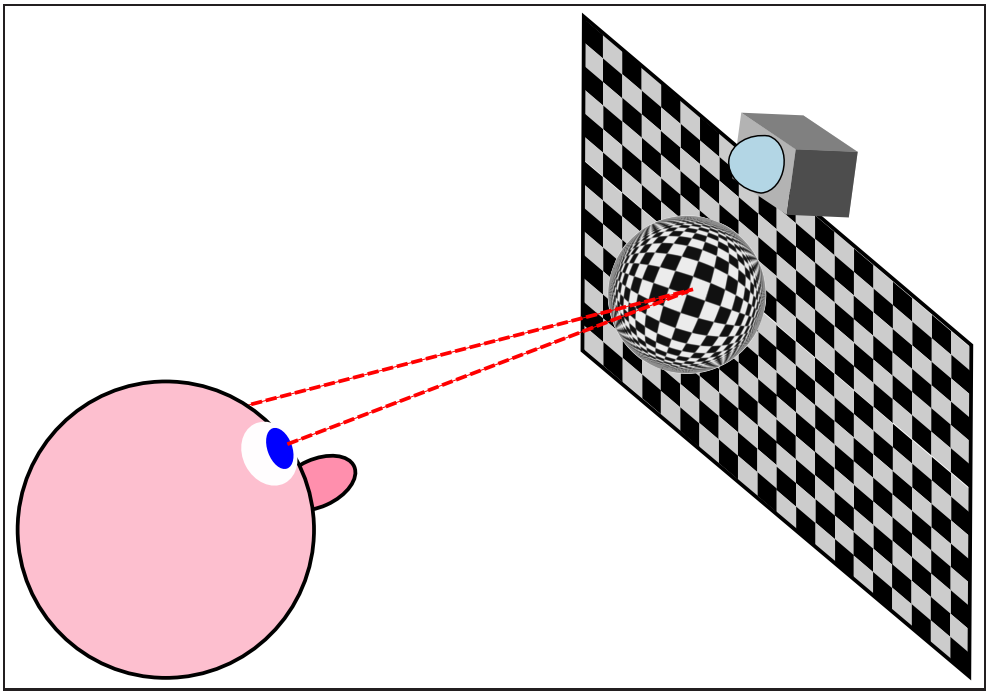
The MLR algorithm could potentially solve the issue of disparity scaling for 3DTV and 3D cinema. In the case of 3DTV, content would be rendered into image slices, broadcast and then the 3D display receiving the signal would be responsible for reprojecting and combining the image slices so as to generate the desirable depth range. We realise that broadcasting a sufficient number of images necessary for good stereoscopic quality may currently be too expensive; however, this problem is likely diminish as technology and bandwidth capabilities improve. In the case of 3D cinema, bandwidth is unlikely to be an issue since there is no need to broadcast any content; a powerful graphics workstation will be responsible for the rendering and driving the projector.

Alternatively, if rendering the stereoscopic imagery on the fly in undesirable (possibly to keep 3D display costs down) the MLR algorithm could be used as high quality 3D movie format for storing master copies. Different 3D movie formats could be rendered as required, before broadcasting, taking into account all three factors: eye separation which may be constrained due to the target audience (e.g. children, adults, ethnic majority), viewing distance and display size which both tend to be coupled, i.e. large displays are usually viewed from a greater range than smaller ones. As a compromise, 3DTV could be broadcast as a few different channels, e.g. for small displays, medium displays, large displays, etc. Effectively we would be using the MLR as a means of portable 3D content. We

believe this would be especially advantageous in the CGI industry because rendering directly from the 3D models is extremely expensive, whereas reprojecting and compositing image slices is relatively cheap.

Currently, using the MLR to generate the image slices from a computer graphics model using points is trivial–triangles should also be possible with further development– and we see great potential for its application in the CGI entertainment industry; however, applying the MLR to real-life content is more challenging. A possible solution would be to use image processing and computer vision techniques to reconstruct a 3D model from the captured images before applying the MLR.

Figure 6.2: The MLR algorithm could potentially be modified to increase or decrease the stereo depth at different rates over the screen depending on where the viewer is looking at.



6.3 Variable screen depth rates

We have discussed how the perceived stereoscopic depth could be controlled arbitrarily by using a depth mapping function and applying greater stereo-resolution to objects of interest; however, another exciting possibility would be to manipulate the disparity at

certain regions of the display screen. One application could be to track the viewer's eyes and to increase the disparity at the focal point but dampen the disparity elsewhere; Figure 6.2 attempts to illustrate this concept. This functionality would be relatively easy to incorporate at the shader stage of the MLR algorithm.

6.4 Occlusion handling

Due to time constraints and the nature of the data, occlusion handling was not implemented in the MLR. However, occlusion handling is possible. As discussed in Chapter 5, the MLR algorithm was implemented in two stages, i.e. two pairs of vertex/fragment shaders; each stage could potentially process occlusion slightly differently. During the first stage of the algorithm the particles in the scene could be sorted by depth and rendered into the texture slices from back to front so that occluding texels are automatically overwritten. Alternatively, an enlarged depth buffer, large enough to fit all the texture slices, could be employed with standard depth testing. The second stage is simpler as we can either render the texture slices from back to front again or render front to back but check each time whether the texel is opaque or transparent and stop accumulating if the texel is opaque.

6.5 Performance optimizations

6.5.1 Early pixel blending termination

Since the particles are rendered into different texture slices based on their depth in the scene, they have in effect been sorted by depth. Therefore, in the compositing stage, starting from the texture slice representing the nearest depth plane, we can terminate the blending calculations early when a pixel becomes saturated or occluded since there is no point in gathering corresponding pixels from the remaining texture slices.

6.5.2 Shared resolution multiview displays

Our implementation of the MLR algorithm currently renders full resolution images for each view and we rely on the multiview display device drivers to display the images in the correct format. However, many multiview displays share the resolution across the views and only require a single image composed from the multiple viewpoint images: the displays use an interleaving pattern to determine which viewpoint each pixel belongs to (see [133] for more details). Therefore, the set of generated images from the MLR would have to be masked and combined before outputting it to a shared resolution multiview display. This is rather wasteful since a lot of pixels are rendered to only be thrown away and extra work is required to combine the images.

One method of increasing performance without changing the implementation is to render each view at the lower per view (shared) resolution and during the masking and combination stage to upsample each view to the full resolution. However, this solution leads to blurring and degraded quality. Fortunately, it is trivial to adapt our program to incorporate the interleaving pattern at the second stage of the algorithm so as to render the final output in a single pass without any masking required. All that is required is to adapt the disparity values for each fragment according to which view it belongs to. Determining which view each fragment belongs to would involve determining the fragments' coordinates and comparing them to a look-up table representing the interleaving pattern. The coordinate of each fragment would simply be calculated from the texture coordinate assigned to it.

6.6 Summary

In this chapter we have presented some of the potential benefits the MLR offers and identified a number of further possible optimizations. To summarise, the potential benefits are:

- Multiple region depth mapping at no extra cost.
- Post-processing (processing on the texture slices, i.e. the original computer graphics model is not required) depth control which could aid in the distribution of mul-

tiview 3D animations to different display formats and sizes.

- Possibility of using the MLR as a high quality stereoscopic file format, since the disparity in the final stereoscopic images can be fully customised and any number of viewpoints generated from the stack of texture slices.

- Capability of outputting multiview images directly into the correct format without the requirement for an extra interleaving and compositing stage using the display device drivers.

- Disparity manipulation at the display screen, which could be used to assign certain regions of the display greater or lower stereoscopic resolution.

Some limitations of the MLR, which require further development and research to overcome, are:

- Rendering support currently only for point data sets–triangles and polygons are not supported.

- Lack of occlusion handling.

- Lack of viewport clipping, i.e. the entire model is assumed to be visible by the camera set-up.

However, even with these limitations we believe the MLR is a useful contribution to the field of stereoscopy, and with further development could find a wider range of applicability than the point data sets the algorithm was originally designed for.

Chapter 7

Conclusions

7.1 Summary

This thesis has shown that it is possible to visualize large scientific point data sets using multiview displays by lowering the number of views required from what the current literature suggests; exploiting perspective coherence across the views; and reducing the number of reprojections according to the resolution of the stereoscopic display.

The optimum number of views required was determined by human factor trials using an experimental design that was adapted from a path tracing task described in [160]. The study is the first to investigate the affects of viewpoint density on depth perception when task performance is taken into consideration rather than aesthetic qualities. While viewpoint densities were observed to have a significant effect on task performance subjects did not appear to significantly benefit when more than 8 views per 10 cm were provided; the results suggest that the optimum number of views may even be as few as four.

The "incremental fragment algorithm" described has almost no set-up cost and therefore provides immediate rendering performance gains for more than two views over the naïve SVR approach. The main idea behind the algorithm was to eliminate redundant stages in a typical splatting rendering pipeline and to take advantage of the stereo-coherence available between the viewpoints. The algorithm assumes that every particle is spherical and translucent, enabling the use of cheap additive blending and eliminating the requirement for dealing with surface normals, depth sorting and occlusion handling. Furthermore, reprojecting the fragments to generate each viewpoint saves on lighting and

perspective projection calculations. While stereoscopic algorithms employing perspective reprojection have been described before, the novelty in this approach is applying the technique solely to point geometry and delaying reprojection and generation of the viewpoints until the fragment stage of the pipeline; this allows greater efficiencies to be made. Unfortunately, the algorithm could not be implemented using the standard OpenGL rendering pipeline or programmable GPU pipeline because of the lack of array indexing; therefore, the results were theoretical in nature.

The second algorithm described, called the MLR, reduces the number of reprojection calculations further by grouping the vertices into their respective voxel planes and calculating the reprojection quantities once per voxel plane instead of for each vertex; the idea was derived from the observation that vertices which are mapped to the same voxel plane are projected onto the display with the same amount of pixel disparity regardless of differences in their scene depth. An advantage of this approach is that flexible multi-region depth mapping can be achieved at little or no extra cost. The programmable pipeline of the GPU was exploited by the MLR implementation for greater performance. Some accuracy is sacrificed and set-up costs are higher; however, rendering performance gains for two views are still achievable for large data sets (2M+ points).

Both algorithms were shown to be potentially much faster than the SVR approach. The incremental fragment algorithm is expected to be at least 2.8 times quicker than the SVR. The MLR is not dependent on the scene complexity but on the pixel and stereoscopic resolution, and therefore is of most advantage in highly complex and detailed scenes; for example, rendering a typical particle data set containing 500K particles for a 9-view 1920x1080 multiview display, the MLR was argued to be approximately 2 times quicker than the incremental fragment approach and shown to be more than 5 times quicker than the SVR. However, in scenarios where the dataset is small, the fragment reprojection algorithm is likely to be more efficient.

7.2 Further work

7.2.1 Human factor trials

Gilson et al. [49] reported that the spatial accuracy of the IS900 head tracker decreased when it was in motion with root mean square (**RMS**) errors of up to 17 mm being observed, although slow movements only induced **RMS** errors of 2.87 mm. This is problematic because these errors may not allow large viewpoint densities to be reproduced accurately e.g. 50 views per 10 cm which requires tracker accuracy of 2 mm. For typical head movements in this experiment, velocities were probably too slow to induce large errors and no anomalies were noticed in the way the graph was updated on the display screen. However, we cannot confirm that tracker error did not affect the results so further investigation on the tracker accuracy is required. If the experiment is to be repeated, an optical tracker would be preferable so as to remove any doubts on the validity of the results.

Also for large amounts of depth the false rotation and flipping effect can be quite noticeable with only 8 views per 10 cm; therefore, further investigation should be carried out on observer tolerances of these artifacts for long durations. However, we suspect observers are unlikely to move their heads repeatedly during longer working periods due to the physical effort required.

Another interesting avenue of research remaining is to investigate in cases where motion parallax has an additive effect with stereopsis on depth perception, whether motion parallax is simply being used to alleviate occlusions or whether it is aiding in the correspondence problem or it is doing both.

7.2.2 Incremental fragment algorithm

Unfortunately the incremental fragment algorithm could not be implemented on the GPU because of the lack of array indexing. This confines the application of the incremental fragment algorithm to software rendering implementations: for example, ray tracing or parallel computing applications which can not usually take advantage of the GPU. Further research would be desirable to determine if there is a potential solution to this problem. Also it would be desirable to incorporate cost reducing methods for viewpoint dependent

lighting. For example, if there are many small splats being rendered with viewpoint dependent lighting enabled, it may be possible to achieve approximated surface rendered results. However, the performance would have to be compared with existing PBR and multiview PBR algorithms such as [76]. Further investigation on the efficiency of various occlusion handling mechanisms available is also required.

7.2.3 MLR

It would also be desirable to investigate the possibility of extending the MLR to render other primitives such as triangles and also to incorporate viewpoint lighting effects. We have already described how to potentially handle occlusion, however, a key question to investigate is whether there would still be any performance gains over the SVR algorithm with this extra functionality. We mentioned that the MLR could produce image inaccuracies of up to 2 pixels; therefore, it would be desirable to quantify how much of an issue (if any) this is by using human trials and to determine if texture filtering helps.

Bibliography

[1] Microsoft directx resource center. http://msdn2.microsoft.com/en-us/xna/aa937781.aspx, (last accessed 01.01.2008).

[2] Microsoft hlsl shaders. http://msdn.microsoft.com/archive/default.asp?url=/archive/en-us/directx9_c_Dec_2005/HLSL_Shaders.asp, (last accessed 01.01.2008).

[3] Nvidia cg toolkit 1.5. http://developer.nvidia.com/object/cg_toolkit.html, (last accessed 01.01.2008).

[4] Opengl & opengl utility specifications. http://www.opengl.org/documentation/specs/, (last accessed 01.01.2008).

[5] Opengl shading language. http://www.opengl.org/documentation/glsl/, (last accessed 01.01.2008).

[6] S. Adelson and C. Hansen. Fast stereoscopic images with Ray-Traced volume rendering. In A. Kaufman and W. Krueger, editors, *1994 Symposium on Volume Visualization*, pages 3–10, 1994.

[7] K. Akeley, S. J. Watt, A. R. Girshick, and M. S. Banks. A stereo display prototype with multiple focal distances. *ACM Transactions on Graphics*, 23:804 – 813, 2004.

[8] T. Annen, W. Matusik, H. Pfister, H. p Seidel, and M. Zwicker. Distributed rendering for multiview parallax displays, 2006.

[9] T. Annen, W. Matusik, H. Pfister, H.-P. Seidel, and M. Zwicker. Distributed rendering for multiview parallax displays. In A. J. Woods, N. A. Dodgson, J. O. Merritt,

M. T. Bolas, and I. E. McDowall, editors, *Stereoscopic Displays and Virtual Reality Systems XIII*, volume 6055 of *Proceedings of SPIE*, pages 231–240. SPIE, Feb. 2006.

[10] T. Balogh, T. Forgács, T. Agócs, O. Balet, E. Bouvier, F. Bettio, E. Gobbetti, and G. Zanetti. A scalable hardware and software system for the holographic display of interactive graphics applications. *EUROGRAPHICS*, 2005.

[11] E. P. Baltsavias. Airborne laser scanning: existing systems and firms and other resources. *ISPRS Journal of Photogrammetry & Remote Sensing*, 54:164–198, 1999.

[12] S. A. Benton, editor. *Selected Papers on Three-Dimensional Displays*, volume MS 162 of *SPIE Milestone*. SPIE Optical Press, 2001.

[13] C. Blakemore. The range and scope of binocluar depth discrimination in man. *Physiology*, 211:599–622, 1970.

[14] O. A. R. Board, D. Shreiner, M. Woo, J. Neider, and T. Davis. *OpenGL Programming Guide: The Official Guide to Learning OpenGL(R), Version 2*. Addison-Wesley Professional; 5 edition, 2005.

[15] D. A. Bowman, E. Kruijff, J. J. LaViola, and I. Poupyrev. *3D User Interfaces: Theory and Practice*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 2004.

[16] M. F. Bradshaw, P. B. Hibbard, A. D. Parton, D. Rose, and K. Langley. Surface orientation, modulation frequency and the detection and perception of depth defined by binocular disparity and motion parallax. *Vision Res.*, 46(17):2636–44, 2006.

[17] M. F. Bradshaw, A. D. Parton, and A. Glennerster. The task-dependent use of binocular disparity and motion parallax information. *Vis. Res.*, 40(27):3725–3734, 2000.

[18] M. L. Braunstein. Motion and texture as sources of slant information. *J. Exp. Psychol.*, 78(2):247–253, 1968.

[19] S. D. Brewster. On the law of visible position in single and binocular vision, and on the representation of solid figures by the union of dissimilar plane pictures on the retina. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 1844.

[20] N. Bruno and J. E. Cutting. Minimodularity and the perception of layout. *Journal of Experimental Psychology: General.*, 117(2):161–170, June 1988.

[21] S. J. Buckley, J. A. Howell, H. D. Enge, and N. A. Kurz. Terrestrial laser scanning in geology: data acquisition, processing and accuracy considerations. *Journal of the Geological Society*, 165:625–638, 2008.

[22] D. R. W. Butts and D. F. McAllister. Implementation of true 3D cursors in computer graphics. In W. E. Robbins, editor, *Three-Dimensional Imaging and Remote Sensing Imaging*, volume 902 of *Proceedings of SPIE*, pages 74–84. SPIE, Jan. 1988.

[23] G. Caillebotte. Jour de pluie à paris. `http://commons.wikimedia.org/wiki/File:Gustave_Caillebotte_-_La_Place_de_l%27Europe,_temps_de_pluie.jpg`, (last accessed 28.10.2008), 1877.

[24] O. M. Castle. *Synthetic Image Generation for a Multiple-View Autostereo Display*. PhD thesis, University of Cambridge, 1995.

[25] S. E. Chen and L. Williams. View interpolation for image synthesis. *Computer Graphics*, 27(Annual Conference Series):279–288, 1993.

[26] W.-S. Chun, J. Napoli, O. S. Cossairt, R. K. Dorval, and D. M. Hall. Spatial 3-D Infrastructure: Display-Independent Software Framework, High-Speed Rendering Electronics, and Several New Displays. In A. J. Woods, M. T. Bolas, J. O. Merritt, and I. E. McDowall, editors, *Stereoscopic Displays and Virtual Reality Systems XII*, volume 5664 of *Proceedings of SPIE*. SPIE, Mar. 2005.

[27] K. J. CiuVreda. *Borish's clinical refraction: Principles and practice*, chapter Accommodation, pupil, and presbyopia, pages 77–120. Philadelphia: Saunders, 1998.

[28] O. S. Cossairt, J. Napoli, S. L. Hill, R. K. Dorval, , and G. E. Favalora. Occlusion-capable multiview volumetric three-dimensional display. *APPLIED OPTICS*, 46(8), Mar. 2007.

[29] R. A. Crain, T. Theuns, C. Dalla Vecchia, V. R. Eke, C. S. Frenk, A. Jenkins, S. T. Kay, J. A. Peacock, F. R. Pearce, J. Schaye, V. Springel, P. A. Thomas, S. D. M. White, and R. P. C. Wiersma. Galaxies-Intergalactic Medium Interaction Calculation –I. Galaxy formation as a function of large-scale environment. *ArXiv e-prints*, June 2009.

[30] B. G. Cumming and G. C. DeAngelis. The physiology of stereopsis. *Annual Review of Neuroscience*, 24(1):203–238, 2001.

[31] E. T. Davis and L. F. Hodges. Human stereopsis, fusion, and stereoscopic virtual environments. pages 145–174, 1995.

[32] F. de Sorbier, V. Nozick, and V. Biri. GPU rendering for autostereoscopic displays. *Fourth International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT)*, 2008.

[33] D. B. Diner and D. H. Fender. Dependence of Panum's fusional area on local retinal stimulation. *J. Opt. Soc. Am. A*, 5(7), July 1988.

[34] DisplaySearch. DisplaySearch Expects Global Flat Panel Display Revenues to Rise 14% in 2006 to $85.5B and Grow 8% in 2007 to $92B Led by TV Panels. `http://www.displaysearch.com/press/?id=1066`, (last accessed 29.07.2007), 2007.

[35] N. A. Dodgson. Variation and extrema of human interpupillary distance. In A. J. Woods, J. O. Merritt, S. A. Benton, and M. T. Bolas, editors, *Stereoscopic Displays and Virtual Reality Systems XI*, volume 5291 of *Proceedings of SPIE*, pages 36–46. SPIE, May 2004.

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

[37] N. A. Dodgson. On the number of viewing zones required for head-tracked autostereoscopic display. In A. J. Woods, N. A. Dodgson, J. O. Merritt, M. T. Bolas,

and I. E. McDowall, editors, *Stereoscopic Displays and Virtual Reality Systems XIII*, volume 6055 of *Proceedings of SPIE*, pages 241–252. SPIE, Feb. 2006.

[38] N. A. Dodgson, J. R. Moore, and S. R. Lang. Multi-view autostereoscopic 3D display. *IEEE Computer*, 38:31–36, 1999.

[39] N. A. Dodgson, J. R. Moore, S. R. Lang, G. J. Martin, and P. M. Canepa. 50-in. time-multiplexed autostereoscopic display. In J. O. Merritt, S. A. Benton, A. J. Woods, and M. T. Bolas, editors, *Stereoscopic Displays and Virtual Reality Systems VII*, volume 3957 of *Proceedings of SPIE*, pages 177–183. SPIE, May 2000.

[40] B. A. Dosher, G. Sperling, and S. Wurst. Tradeoffs between stereopsis and proximity luminance covariance as determinants of perceived 3D structure. *Vis. Res.*, 26(6):973–990, 1986.

[41] F. H. Durgin and D. R. Proffitt. Comparing depth from motion with depth from binocular disparity. *J. Exp. Psychol. Hum. Percept. Perform.*, 21(3):679–699, June 1995.

[42] T. Endo, Y. Kajiki, T. Honda, and M. Sato. A cylindrical 3-D video display observable from all directions. In J. O. Merritt, S. A. Benton, A. J. Woods, and M. T. Bolas, editors, *Stereoscopic Displays and Virtual Reality Systems VII*, volume 3957 of *Proceedings of SPIE*, pages 225–233. SPIE, 2000.

[43] J. Faubert. *Three-Dimesional Video and Display: Devices and Systems*, chapter Motion parallax, stereoscopy, and the perception of depth: Practical and theoretical issues, pages 168–191. SPIE Optical Engineering Press, 2001.

[44] C. Fehn. A 3D-TV approach using depth-image-based rendering (DIBR). In *Proceedings of 3rd IASTED Conference on Visualization, Imaging, and Image Processing*, pages 482–487, Benalmdena, Spain, September 2003.

[45] I. Fine and R. A. Jacobs. Modeling the combination of motion, stereo, and vergence angle cues to visual depth. *Neural Comput.*, 11(6):1297–1330, 1999.

[46] f. Fir0002. Image:moire on parrot feathers.jpg. http://commons.wikimedia.org/wiki/Image:Moire_on_parrot_feathers.jpg, (last accessed 28.10.2008), undated.

[47] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes. *Computer Graphics: Principles and Practice in C (International Edition)*. Addison Wesley, 1995.

[48] P. S. Foresman. Image:stereoscope (psf).png. http://commons.wikimedia.org/wiki/Image:Stereoscope_(PSF).png, (last accessed 28.10.2008), undated.

[49] S. J. Gilson, A. W. Fitzgibbon, and A. Glennerster. Quantitative analysis of accuracy of an inertial/acoustic 6dof tracking system in motion. *J. Neurosci. Methods.*, 154(1-2):175–82, 2006.

[50] T. Girtin. Die kathedrale von Durham und die brücke, vom fluß Wear aus gesehen. http://commons.wikimedia.org/wiki/Image:Thomas_Girtin_002.jpg, (last accessed 28.10.2008), 1799.

[51] G.J.Woodgate and D.Ezra. European patent application serial no. ep 0 625 861. Filed 20 May 1993.

[52] A. Glennerster, B. J. Rodgers, and M. F. Bradshaw. Stereoscopic depth constancy depends on the subject's task. *Vis. Res.*, 36(21):3441–3456, Nov. 1996.

[53] E. B. Goldstein. *Sensation and Perception sixth edition*. Wadsworth, 2002.

[54] D. G. Green, M. K. Powers, and M. S. Banks. Depth of focus, eye size and visual acuity. *Vision Research*, 20:827–835, 1979.

[55] N. Greene and M. Kass. Approximate visibility with environment maps. Technical report 41, 1993, Apple Computer, Inc, 1993.

[56] P. Gurram, E. Saber, and H. Rhody. A novel triangulation method for building parallel-perspective stereo mosaics. In *Stereoscopic Displays and Applications XVIII*, pages 3–10, 2007.

[57] G.Woodgate, R. Moseley, D. Ezra, and N. Holliman. Autosterescopic display us pat. no. 6.055.013. April 2000, priority Feb. 1997.

[58] M. Halle. Holographic stereograms as discrete imaging systems, 1994.

[59] M. Halle. Autostereoscopic displays and computer graphics. *Computer Graphics, ACM SIGGRAPH*, 31(2):58–62, May 1997.

[60] M. Halle. Multiple viewpoint rendering. *Computer Graphics*, 32(Annual Conference Series):243–254, 1998.

[61] J. Harrold and G. J. Woodgate. Autostereoscopic display technology for mobile 3DTV applications. In A. J. Woods, N. A. Dodgson, J. O. Merritt, M. T. Bolas, and I. E. McDowall, editors, *Stereoscopic Displays and Virtual Reality Systems XIV*, volume 6490 of *Proceedings of SPIE*. SPIE, 2007.

[62] J. Hasselgren and T. Akenine-Moller. An efficient multiview rasterization architecture. *Eurographics Workshop/ Symposium on Rendering*, pages 61–72, 2006.

[63] W. Hess. Stereoscopic picture. U.S. Patent No. 1,128,979, 1915.

[64] N. Holliman. Mapping perceived depth to regions of interest in stereoscopic images. volume 5291, pages 117–128. SPIE, 2004.

[65] N. Holliman. Smoothing region boundaries in variable depth mapping for real-time stereoscopic images. In A. J. Woods, M. T. Bolas, J. O. Merritt, and I. E. McDowall, editors, *Stereoscopic Displays and Virtual Reality Systems XII*, volume 5664 of *Proceedings of SPIE*, pages 281–292. SPIE, Mar. 2005.

[66] N. S. Holliman. *Handbook of Optoelectronics*, volume 2, chapter Three-Dimensional Display Systems. Taylor & Francis, May 2006.

[67] T. Honda, Y.Kajiki, K. Susami, T. Hamaguchi, T. Endo, T. Hatada, and T. Fujii. *Three-Dimesional Video and Display: Devices and Systems*, chapter Three-dimensional display technologies satisfying ”super multivew condition”, pages 230–246. SPIE Optical Engineering Press, 2000.

[68] M. Hopf and T. Ertl. Hierarchical splatting of scattered data. In *VIS '03: Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, page 57, Washington, DC, USA, 2003. IEEE Computer Society.

[69] M. Hopf, M. Luttenberger, and T. Ertl. Hierarchical splatting of scattered 4D data. *IEEE Comput. Graph. Appl.*, 24(4):64–72, 2004.

[70] A. M. Horwood, J. E. Turner, S. M. Houston, and P. M. Riddell. Variations in accommodation and convergence responses in a minimally controlled photorefractive setting. *Optometry and Visison Science*, 78(11):791–804, 2001.

[71] K. Hosokawa, S. Ohtsuka, and T. Sato. Depth perception from intermittent motion parallax stimuli. *J. Vis.*, 5(8):729–729, 9 2005.

[72] H. J. Howard. A test for the judgement of distance. *Am. J. Ophthal.*, 2:656–675, 1919.

[73] I. P. Howard. *Seeing in Depth*, volume 1. I Porteous, Toronto, 2002.

[74] J. Hsu, Z. Pizlo, D. Chelberg, C. Babbs, and E. Delp. Issues in the design of studies to test the effectiveness of stereo imaging, 1996.

[75] K. Huang, J. Yuan, C. Tsai, and W. Hsueh. Crosstalk issue affecting stereopsis in stereoscopic display. In A. J. Woods, M. T. Bolas, and J. O. M. S. A. Benton, editors, *Stereoscopic Displays and Virtual Reality Systems X*, volume 5006 of *Proceedings of SPIE*. SPIE, May 2003.

[76] 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 Press.

[77] W. A. IJsselsteijn, H. de Ridder, J. Freeman, S. E. Avons, and D. Bouwhuis. Effects of stereoscopic presentation, image motion and screen size on subjective and objective corroborative measures of presence. *Presence: Teleoperators and Virtual Environments.*, 10(3):298–311, 2001.

[78] F. E. Ives. A novel stereogram. *Franklin Institue*, 153:51–52, 1902.

[79] E. Jin, M. E. Miller, S. Endrikhovski, and C. D. Cerosaletti. Creating a comfortable stereoscopic viewing experience: Effects of viewing distance and field of view on fusional range. In A. J. Woods, M. T. Bolas, J. O. Merritt, and I. E. McDowall, editors, *Stereoscopic Displays and Virtual Reality Systems XII*, volume 5664 of *Proceedings of SPIE*, pages 10–21. SPIE, Mar. 2005.

[80] A. Johnston and P. J. Passmore. Independent encoding of surface orientation and surface curvature. *Vision Research*, 34(22):3005–3012, 1994.

[81] E. B. Johnston. Systematic distortions of shape from stereopsis. *Vision Res.*, 31(7/8):1351–1360, 1991.

[82] E. B. Johnston, B. G. Cumming, and M. S. Landy. Integration of stereopsis and motion shape cues. *Vision Research.*, 34(17):2259–2275, Sep. 1994.

[83] G. R. Jones, D. Lee, N. S. Holliman, and D. Ezra. Controlling perceived depth in stereoscopic images. volume 4297, pages 42–53. SPIE, 2001.

[84] B. Julesz. Binocular depth perception of computer-generated patterns. *Bell System Technical Journal*, 1960.

[85] Y. Kajiki, H. Yoshikawa, and T. Honda. Three-dimensional display with focused light array. In S. A. Benton, editor, *Practical Holography X*, volume 2652 of *Proceedings of SPIE*, pages 106–116. SPIE, 1996.

[86] Y. Kajiki, H. Yoshikawa, and T. Honda. Hologram-like video images by 45-view stereoscopic display. In *Stereoscopic Displays and Virtual Reality Systems IV*, 1997.

[87] R. Kaptein and I. Heynderickx. Effect of crosstalk in multi-view autostereoscopic 3D displays on perceived image quality. *SID 07 DIGEST*, pages 1220–1223, 2007.

[88] S. Knorr, E. Imre, B. Ozkalayci, A. A. Alatan, and T. Sikora. A modular scheme for 2D/3D conversion of TV broadcast. In *3DPVT '06: Proceedings of the Third International Symposium on 3D Data Processing, Visualization, and Transmission*

(3DPVT'06), pages 703–710, Washington, DC, USA, 2006. IEEE Computer Society.

[89] J. J. Koenderink and A. J. van Doorn. Affine structure from motion. *J. Opt. Soc. Am. A.*, 8(2):377–385, Feb. 1991.

[90] J. Konrad. Enhancement of viewer comfort in stereoscopic viewing: parallax adjustment, 1999.

[91] F. L. Kooi and A. Toet. Additive and subtractive transparent depth displays. In J. G. Verly, editor, *Enhanced and Synthetic Vision 2003*, volume 5081 of *Proceedings of SPIE*, pages 58–65. SPIE, Sept. 2003.

[92] F. L. Kooi and A. Toet. Visual comfort of binocular and 3D displays. *Proc SID*, 25:99–108, 2004.

[93] L. L. N. Laboratory. File:rayleigh-taylor instability.jpg. `http://commons.wikimedia.org/wiki/File:Rayleigh-Taylor_instability.jpg`, (last accessed 05.10.2009), 2007.

[94] W. S. Lages, C. cordeiro, and D. Guedes. A parallel multi-view rendering architecture. In C. R. Jung and M. Walter, editors, *Brazilian Symposium on Computer Graphics and Image Processing*, pages 270–277, Los Alamitos, Oct. 12–15, 2008 2008. IEEE Computer Society.

[95] M. T. M. Lambooij, W. A. IJsselsteijn, and I. Heynderickx. Visual discomfort in stereoscopic displays: a review. In *Stereoscopic Displays and Virtual Reality Systems XIV*, volume 6490 of *Proceedings of SPIE*. SPIE, Feb. 2007.

[96] M. S. Landy and H. Kojima. Ideal cue combination for localizing texture-defined edges. *J. Opt. Soc. Am. A. Opt. Image. Sci. Vis.*, 18(9):2307–2320, Sep. 2001.

[97] M. S. Landy, L. T. Maloney, E. B. Johnston, and M. Young. Measurement and modeling of depth cue combination: in defense of weak fusion. *Vis. Res.*, 35(3):389–412, Feb. 1995.

[98] B. Lee, H. Hong, J. Park, H. Park, H. Shin, and I. Jung. Multi view autostereoscopic display of 36 view using an ultra-high resolution LCD. In A. J. Woods, N. A. Dodgson, J. O. Merritt, M. T. Bolas, and I. E. McDowall, editors, *Stereoscopic Displays and Virtual Reality Systems XIV*, volume 6490 of *Proceedings of SPIE*. SPIE, 2007.

[99] M. Levoy and T. Whitted. The use of points as a display primitive, 1985.

[100] J. D. Lewis, C. M. Verber, and R. B. McGhee. A true three-dimensional display. *IEEE Transactions on Electron Devices*, 1971.

[101] L. Lipton. *Foundations of the Stereoscopic Cinema: A Study in Depth*. Van Nostrand Reinhold, New York, 1982.

[102] L. Lipton. *Stereographics Developers Handbook*. Stereographics Corporation, 1997.

[103] J. Liu, S. Pastoor, K. Seifert, and J. Hurtienne. Three-dimensional pc: toward novel forms of human-computer interaction. In *Three-Dimensional Video and Display: Devices and Systems*, Proceedings of SPIE. SPIE, 2000.

[104] D. T. Marshall. *The Exploitation of Image Construction Data and Temporal/Image Coherence in Ray Traced Animation*. PhD thesis, The University of Texas at Austin, 2001.

[105] K. Mashitani, G. Hamagishi, M. Higashino, T. Ando, and S. Takemoto. Step barrier system multiview glassless 3D display. In A. J. Woods, J. O. Merritt, S. A. Benton, and M. T. Bolas, editors, *Stereoscopic Displays and Virtual Reality Systems XI*, volume 5291 of *Proceedings of SPIE*, pages 265–272. SPIE, May 2004.

[106] G. Mather. The use of image blur as a depth cue. *Perception*, 26:1147–1158, 1997.

[107] W. Matusik and H. Pfister. 3D TV: A scalable system for real-time acquisition, transmission, and autostereoscopic display of dynamic scenes. 2004. Mitsubishi Electric Research Laboratories, Cambridge, MA.

[108] D. F. McAllister, editor. *Stereo Computer Graphics and Other True 3D Technologies*. Princeton University Press, Princeton, 1993.

[109] J. O. Merritt and R. E. Cole. Interaction between binocular and monocular depth cues in teleoperation task performance. *SID 92 Digest*, 1992.

[110] T. Mitsuhashi. Evaluation of stereoscopic picture quality with ccf. *Ergonomics*, 39(11):1244–1356, 1996.

[111] M. Mon-Williams, J. P. Wann, and S. Rushton. Binocular vision in a virtual world: visual deficits following the wearing of a head-mounted display. *Ophthalmic. Physiol. Opt.*, 13(4):387–391, Oct. 1993.

[112] J. R. Moore, N. A. Dodgson, A. R. Travis, and S. R. Lang. Time-multiplexed color autostereoscopic display. In S. S. Fisher, J. O. Merritt, and M. T. Bolas, editors, *Stereoscopic Displays and Virtual Reality Systems III*, volume 2653 of *Proceedings of SPIE*, pages 10–19. SPIE, Apr. 1996.

[113] H. Nakanuma, H. Kamei, and Y. Takaki. Natural 3D display with 128 directional images used for human-engineering evaluation. volume 5664, pages 28–35. SPIE, 2005.

[114] K. N. Ogle and A. Prangen. Observations on vertical divergences and hyperphorias. *Arch Ophthal*, 49(313–334), 1953.

[115] T. Okoshi. *Three-Dimensional Imaging Techniques*. Academic Press, 1976.

[116] M. E. Ono, J. Rivest, and H. Ono. Depth perception as a function of motion parallax and absolute-distance information. *Journal of Experimental Psychology: Human Perception and Performance*, 12:331–337, 1986.

[117] I. Oruç, L. T. Maloney, and M. S. Landy. Weighted linear cue combination with possibly correlated error. *Vis. Res.*, 43:2451–2468, 2003.

[118] T. V. Papthomas, J. A. Schiavone, and B. Julesz. Stereo animation for very large data bases: Case study - meteorology. *IEEE Computer Graphics and Applications*, 7(9):18–27, 1987.

[119] S. Pastoor and K. Schenke. Subjective assessments of the resolution of viewing directions in a multi-viewpoint 3D TV system. *Proc SID.*, 30(3):217–222, 1989.

[120] E. Peli. Health and safety issues with head-mounted displays(hmd). IDW (International Display Workshop) 1996 in Kobe, pages 493-496, 1996.

[121] K. Perlin. Displayer and method for displaying us 6239830. May 2001, (filed May 1999).

[122] K. Perlin, S. Paxia, and J. S. Kollin. An autostereoscopic display. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 319–326, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[123] K. Perlin, C. Poultney, J. S. Kollin, D. T. Kristjansson, and S. Paxia. Recent advances in the NYU autostereoscopic display. In A. J. Woods, M. T. Bolas, J. O. Merritt, and S. A. Benton, editors, *Stereoscopic Displays and Virtual Reality Systems VIII*, volume 4297 of *Proceedings of SPIE*, pages 196–203. SPIE, jun 2001.

[124] T. Peterka. *Dynallax: Dynamic Parallax Barrier Autostereoscopic Display*. PhD thesis, University of Illinois at Chicago, 2007.

[125] J. Pfautz. *Depth perception in computer graphics*. PhD thesis, Trinity College, University of Cambridge, May 2000.

[126] E. Reinhard, G. Ward, S. Pattanaik, and P. Debevec. *High Dynamic Range Imaging: Acquisition, Display, and Image-Based Lighting (The Morgan Kaufmann Series in Computer Graphics)*. Morgan Kaufmann, 2005.

[127] I. Relke, M. Klippstein, and B. Riemann. Assessment and improvement of the stereo-image visualization on X3D technologies 3D displays. In A. J. Woods, J. O. Merritt, S. A. Benton, and M. T. Bolas, editors, *Stereoscopic Displays and Virtual Reality Systems XI*, volume 5291, pages 204–211, May 2004.

[128] I. Relke and B. Riemann. Three-dimensional Multiview Large Projection System. In A. J. Woods, M. T. Bolas, J. O. Merritt, and I. E. McDowall, editors, *Stereo-*

scopic Displays and Virtual Reality Systems XII, volume 5664 of *Proceedings of SPIE*, pages 167–174. SPIE, Mar. 2005.

[129] W. Richards. Structure from stereo and motion. *J Opt Soc Am A.*, 2(2):343–349, Feb. 1985.

[130] B. J. Rogers and M. E. Graham. Motion parallax as an independent cue for depth perception. *Perception*, 8:125–134, 1979.

[131] D. Runde. How to realize a natural image reproduction using stereoscopic displays with motion parallax. *IEEE Trans. Circuits Syst. Video Techn.*, 10(3):376–386, 2000.

[132] S. Rusinkiewicz and M. Levoy. QSplat: A multiresolution point rendering system for large meshes. *SIGGRAPH*, 2000.

[133] A. Schmidt and A. Grasnick. Multi-viewpoint Autostereoscopic Displays from 4D-Vision. In A. J. Woods, J. O. Merritt, S. A. Benton, and M. T. Bolas, editors, *Stereoscopic Displays and Virtual Reality Systems IX*, volume 4660 of *Proceedings of SPIE*, pages 212–221. SPIE, May 2002.

[134] B. Schneider, G. Moraglia, and A. Jepson. Binocular Unmasking: An Analog to Binaural Unmasking? *Science*, 243:1479–1481, Mar. 1989.

[135] J. L. Semmlow and D. Heerema. The role of accommodative convergence at the limits of fusional vergence. *Investigative ophthalmology & visual science*, 18(9):970–6, Sep 1979.

[136] J. Shade, S. Gortler, L. wei He, and R. Szeliski. Layered depth images. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 231–242, New York, NY, USA, 1998. ACM.

[137] K. Sharma and A. S. Abdul-Rahim. Vertical fusion amplitude in normal adults. *Am J Ophthal*, 114(636–637), 1992.

[138] M. Siegel. Perceptions of crosstalk and the possibility of a zoneless autostereoscopic display. In A. J. Woods, M. T. Bolas, J. O. Merritt, and S. A. Benton,

editors, *Stereoscopic Displays and Virtual Reality Systems VIII*, volume 4297 of *Proceedings of SPIE*, pages 34–41. SPIE, June 2001.

[139] M. Siegel and L. Lipton. Virtual voxel: a quantitative figure of merit for autostereoscopic display technology and implementation. In *Stereoscopic Displays and Virtual Reality Systems XI*, volume 5291 of *Proceedings of SPIE*. Electronic Imaging Science and Technology, SPIE, 2004.

[140] M. Siegel, Y. Tobinaga, and T. Akiya. Kinder gentler stereo. In *Stereoscopic Displays and Virtual Reality Systems VI*, volume 3639A of *Proceedings of SPIE*, pages 18 – 27. SPIE, Jan. 1999.

[141] M. Slater. How colorful was your day? why questionnaires cannot assess presence in virtual environments. *Presence*, 13(4):484–493, 2004.

[142] R. L. Sollenberger and P. Milgram. Effects of stereoscopic and rotational displays in a three-dimensional path-tracing task. *Human Factors.*, 35(3):483–499, Sep. 1993.

[143] F. Speranza, W. J. Tam, T. Martin, and L. Stelmach. Perceived smoothness of viewpoint transition in multi-viewpoint stereoscopic displays. volume 5664, pages 72–82. SPIE, 2005.

[144] V. Springel. The cosmological simulation code gadget-2. *MON.NOT.ROY.ASTRON.SOC.*, 364:1105, 2005.

[145] V. Springel, S. D. M. White, A. Jenkins, C. S. Frenk, N. Yoshida, L. Gao, J. Navarro, R. Thacker, D. Croton, J. Helly, J. A. Peacock, S. Cole, P. Thomas, H. Couchman, A. Evrard, J. Colberg, and F. Pearce. Simulations of the formation, evolution and clustering of galaxies and quasars. *Nature*, 435:629–636, June 2005.

[146] V. Springel, N. Yoshida, and S. D. M. White. Gadget: A code for collisionless and gasdynamical cosmological simulations. *NEW ASTRON.*, 6:79, 2001.

[147] J. Stewart, E. P. Bennett, and L. McMillan. Pixelview: a view-independent graphics rendering architecture. In *HWWS '04: Proceedings of the ACM SIGGRAPH/EU-*

ROGRAPHICS conference on Graphics hardware, pages 75–84, New York, NY, USA, 2004. ACM.

[148] T. Takeda, K. Hashimoto, N. Hiruma, and Y. Fukui. Characteristics of accommodation toward apparent depth. *Vision Research*, 39:2087–2097, 1999.

[149] J. S. Tittle and M. L. Braunstein. Recovery of 3-D shape from binocular disparity and structure from motion. *Percept. Psychophys.*, 54(2):157–169, 1993.

[150] J. T. Todd and J. F. Norman. The visual perception of smoothly curved surfaces from minimal apparent motion sequences. *Percept. Psychophys.*, 50(6):509–523, Dec. 1991.

[151] J. Torborg and J. T. Kajiya. Talisman: Commodity realtime 3D graphics for the PC. In *SIGGRAPH*, pages 353–363, 1996.

[152] A. R. L. Travis. Autostereoscopic 3-D display. *Applied Optics*, 29:4341–4343, Oct. 1990.

[153] A. R. L. Travis and S. R. Lang. The design and evaluation of a CRT-based autostereoscopic 3-D display. In *Proceeding of the Society for Information Display*, volume 32, pages 279–283, 1991.

[154] C. W. Tyler. *Duane's Foundations of Clinical Ophthalmology*, volume 2, chapter Binocular Vision. J.B. Lippincott Co.: Philadelphia, 2004.

[155] N. A. Valyus. *Stereoscopy*. The Focal Press, London and New York, 1966.

[156] C. vanBerkel and J. Clarke. Characterisation and Optimisation of 3D-LCD Module Design. In S. S. Fisher, J. O. Merritt, and M. T. Bolas, editors, *Stereoscopic Displays and Virtual Reality Systems IV*, volume 3012 of *Proceedings of SPIE*. SPIE, 1997.

[157] C. Vázquez. View generation for 3D-TV using image reconstruction from irregularly spaced samples. In *Stereoscopic Displays and Applications XVIII*, 2007.

[158] A. Vetro, W. Matusik, H. Pfister, and J. Xin. Coding approaches for end-to-end 3D TV systems. Mitsubishi Electric Research Laboratories, Cambridge, MA, Dec. 2004.

[159] N. Wang. Let there be clouds. *CMP Media LLC Game Developer*, page 34, 2004.

[160] C. Ware and G. Franck. Evaluating stereo and motion cues for visualizing information nets in three dimensions. *ACM Trans. Graph.*, 15(2):121–140, 1996.

[161] C. Ware, C. Gobrecht, and M. Paton. Algorithm for dynamic disparity adjustment. In S. S. Fisher, J. O. Merritt, and M. T. Bolas, editors, *Stereoscopic Displays and Virtual Reality Systems II*, volume 2409 of *Proceedings of SPIE*, pages 150–156. SPIE, Mar. 1995.

[162] C. Ware and P. Mitchell. Reevaluating stereo and motion cues for visualizing graphs in three dimensions. In *APGV '05: Proceedings of the 2nd symposium on Applied perception in graphics and visualization*, pages 51–58, New York, NY, USA, 2005. ACM Press.

[163] Z. Wartell. *Stereoscopic Head-Tracked Displays: Analysis and Development of Display Algorithms*. PhD thesis, Georgia Institute of Technology, 2001.

[164] W. R. Watkins, G. D. Heath, M. D. Phillips, J. M. Valeton, and A. Toet. Search and target acquisition: single line of sight versus wide baseline stereo. *Optical Engineering*, 40:1914–1927, Sept. 2001.

[165] L. Westover. Interactive volume rendering. *Proceedings of the Chapel Hill Workshop on Volume Visualization*, 1989.

[166] C. Wheatstone. Contributions to the physiology of vision.-part the first. on some remarkable and hitherto unobserved, phenomena of binocular vision. *Philosophical Transactions of the Royal Society of London*, 1838.

[167] S. P. Williams and R. V. Parrish. New computational control techniques and increased understanding for stereo 3-D displays. In S. S. Fisher and J. O. Merritt, editors, *Stereoscopic Displays and Applications*, volume 1256 of *Proceedings of SPIE*, pages 73–82. SPIE, 1990.

[168] G. Woodgate, J. Harrold, M. Jacobs, R. Moseley, and D. Ezra. Flat panel autostereoscopic displays - characterisation and enhancement. In *Stereoscopic displays and virtual reality systems VII*, volume 3957 of *Proceedings of SPIE*. SPIE, 2000.

[169] G. J. Woodgate, D. Ezra, J. Harrold, N. S. Holliman, G. R. Jones, and R. R. Moseley. Observer-tracking autostereoscopic 3D display systems. In S. S. Fisher, J. O. Merritt, and M. T. Bolas, editors, *Stereoscopic Displays and Virtual Reality Systems IV*, volume 3012 of *Proceedings of SPIE*, pages 187–198. SPIE, May 1997.

[170] A. Woods, T. Docherty, and R. Koch. Image Distortions in Stereoscopic Video Systems. In J. O. Merritt and S. S. Fisher, editors, *Stereoscopic Displays and Applications IV*, volume 1915 of *Proceedings of SPIE*, pages 36–48. SPIE, 1993.

[171] M. Wopking. Viewing comfort with stereoscopic pictures: An experimental study on the subjective effects of disparity magnitude and depth of focus. *Journal of the Society for Information Display*, 3(3):101–103, 1995.

[172] J. C. Yang, M. Everett, C. Buehler, and L. Mcmillan. A real-time distributed light field camera. In *Eurographics Association*, pages 77–86, 2002.

[173] S. Yano, M. Emotoa, and T. Mitsuhash. Two factors in visual fatigue caused by stereoscopic hdtv images. *Displays*, 25(4):141–150, 2004.

[174] S. Yano, S. Ide, T. Mitsuhashi, and H. Thwaites. A study of visual fatigue and visual comfort for 3D HDTV/HDTV image. *Displays*, 23(4):191–201(11), 2002.

[175] Y. Yeh and L. D. Silverstein. Limits of fusion and depth judgment in stereoscopic color displays. *Hum. Factors*, 32(1):45–60, 1990.

[176] Y. Yeh and L. D. Silverstien. Limits of fusion and depth judgement in stereoscopic colour displays. *Human Factors*, 32(1):45–60, 1990.

[177] M. J. Young, M. S. Landy, and L. T. Maloney. A perturbation analysis of depth perception from combinations of texture and motion cues. *Vis. Res.*, 33(18):2685–2696, Dec. 1993.

[178] Z. Zhu, A. Hanson, and E. Riseman. Generalized parallel-perspective stereo mosaics from airborne video. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 26, pages 226–237, 2004.

Appendix A

Incremental fragment algorithm

A.1 Vertex Shader

```
uniform float atn, S, coolestT, tRange, opacityScale, sPix, lPix;
uniform vec3 coolestCol, hotestCol;
attribute float vertexT, density;
varying float temp;

void main() {
 vec4 vec, homeye, eye;
 float t;
 vec4 tCol = gl_Color;
 // relative coords to absolute coords
 homeye = gl_ModelViewMatrix * gl_Vertex;
 gl_Position = gl_ProjectionMatrix * homeye;
 eye = homeye / homeye.w;
 // point size calculation
 t = inversesqrt(atn*dot(eye.xyz, eye.xyz));
 // scale size according to density
 t = t*((lPix - (tCol.w * lPix)) + sPix) * S;
 // clamp pixel size
 t = clamp (t, sPix, lPix);
 gl_PointSize = t;
```

```
 // convert temperature into range of [0.0, 1.0]
 temp = (vertexT - coolestT) / tRange;
 t = tColr.w * opacityScale;
 t = clamp (t, 0.0, 1.0);
 // compute colour based on temperature
 vec3 shade = mix(coolestCol, hotestCol,temp);
 gl_FrontColor = vec4(shade, t);
}
```

Appendix B

Multi-layered rendering algorithm

B.1 Vertex Shader 1

```
// scales all the calculated opacities with this value
uniform float opacityScale;
// this values specifies the smallest allowed opacity
uniform float smallestOpacity;
// smallest pixel size allowed for each data point
uniform float smallestPixelSize;
// largets pixel size allowed
uniform float largestPixelSize;
uniform int no_of_horizontal_slices;
uniform float SceneNearPlane;
uniform float SceneFarPlusNearPlane;
// coolest temperature particles are shaded with this colour
uniform vec3 coolestColour;
// hottest temperature particles are shaded with this colour
uniform vec3 hotestColour;
// the size of each pixel on the scene viewing window
uniform float scenePixelWidth;
// the camera separation
uniform float camSep;
uniform int no_of_negative_slices;
```

```
uniform float sceneCameraDistance;
uniform float pixelSizeDifference;
uniform int no_slices_per_texture;
uniform float f_no_of_horizontal_slices;
uniform float f_no_of_horizontal_slices_minus_one;

attribute float vertexTemp;
attribute float density;

varying float texUnit;

void main()
{
  vec4 vec, homeye, eye;
  float tmp;
  vec4 tmpColour = gl_Color;
  int textureUnit; // which texture unit is being rendered to

  // transform vertices so viewing frustum is placed at the
  // origin and looking down the negative z axis
  homeye = gl_ModelViewMatrix * gl_Vertex;

  // calculate size of point based on how far it is from viewer
  // first convert density range from 0-1 into smallest and
  // largest pixel size range.
  float size = density * pixelSizeDifference + smallestPixelSize;

  // project particle size onto near viewing plane
  float negative_homeye = -1.0*homeye.z;
  size = (SceneNearPlane * size) / negative_homeye;
  float minProjection = (SceneNearPlane * smallestPixelSize) /
    negative_homeye;
```

```
// convert projection sizes into range of [0.0, 1.0]
size = (size - minProjection) /
  (largestPixelSize - minProjection);
// now expand scale to required minimum and maximum pixel size
size = size*pixelSizeDifference+smallestPixelSize;

// calculate which slice the point should be in.
// slices are numbered from 1 onwards with slice 1 being closest
// voxel plane to the viewer ie coming out of the monitor.
float Zrel = negative_homeye/sceneCameraDistance;
float fSlice_no = ((camSep * (Zrel - 1.0)) /
  (Zrel * scenePixelWidth));

// project the points using perspective projection
vec = gl_ProjectionMatrix * homeye;
// each vector or point is now in clip coordinates.

// Each slice is rendered to either 4 different texture units
// within each texture unit the slices are arranged row order
// so the 2nd slice is on the 1st row and 1 across in the
// texture unit round and cast the fSlice_no to the nearest int.
int slice_no = int( sign(fSlice_no) *
  floor( abs(fSlice_no) + 0.5 ) ) + no_of_negative_slices;

textureUnit  = (slice_no - 1) / no_slices_per_texture;

slice_no = slice_no - (textureUnit * no_slices_per_texture);
texUnit = float(textureUnit);

// Each slice is rendered as a small texture within the larger
// texture depending on the points depth it will be rendered
// in a different texture position.
int row = (slice_no - 1) / no_of_horizontal_slices;
```

```
    float fRow = float(row);
    int column = slice_no - (row * no_of_horizontal_slices+1);
    float fColumn = float(column);

    // All points are scattered throughout the entire viewing
    // frustum. After perspective projection the particles must
    // be squashed into a bounding box a fourth of the current
    // size so that there is space to render 16 slices (Assuming
    // 64 slices need to be rendered and 4 texture units are
    // avaiable).
    vec.x = vec.x / f_no_of_horizontal_slices -
      ( (vec.w * f_no_of_horizontal_slices_minus_one) /
        f_no_of_horizontal_slices) +
        (fColumn * ((vec.w * 2.0) / f_no_of_horizontal_slices));
    vec.y = vec.y / f_no_of_horizontal_slices +
      ( (vec.w * f_no_of_horizontal_slices_minus_one) /
        f_no_of_horizontal_slices) -
        (fRow * ((vec.w * 2.0) / f_no_of_horizontal_slices));

    gl_Position = vec;
    gl_PointSize = size;

    // temperature is in the range of [0.0, 1.0]
    tmp = tmpColour.w * opacityScale;
    tmp = clamp(tmp, smallestOpacity, 1.0);

    vec3 shade = mix(coolestColour, hotestColour, vertexTemp);
    gl_FrontColor = vec4(shade,tmp);
}
```

B.2 Fragment Shader 1

```
varying float texUnit; // texture unit the fragment belongs to.
```

```
void main()
{
  vec4 discardFrag = vec4(0.0,0.0,0.0,0.0);
  // There is some inprecision with the varying floats
  // the value should either be 0, 1, 2 or 3
  // however it varies very slightly, hence the
  // small range check.
  if(texUnit >= 0.0 && texUnit < 0.999)
  {
    gl_FragData[0] = gl_Color;
    gl_FragData[1] = discardFrag;
    gl_FragData[2] = discardFrag;
    gl_FragData[3] = discardFrag;
  }
  else if(texUnit >= 0.999 && texUnit < 1.999)
  {
    gl_FragData[0] = discardFrag;
    gl_FragData[1] = gl_Color;
    gl_FragData[2] = discardFrag;
    gl_FragData[3] = discardFrag;
  }
  else if(texUnit >= 1.999 && texUnit < 2.999)
  {
    gl_FragData[0] = discardFrag;
    gl_FragData[1] = discardFrag;
    gl_FragData[2] = gl_Color;
    gl_FragData[3] = discardFrag;
  }
  else if(texUnit >= 2.999 && texUnit < 3.999)
  {
    gl_FragData[0] = discardFrag;
    gl_FragData[1] = discardFrag;
```

```
    gl_FragData[2] = discardFrag;
    gl_FragData[3] = gl_Color;
  }
  else
  {
    gl_FragData[0] = discardFrag;
    gl_FragData[1] = discardFrag;
    gl_FragData[2] = discardFrag;
    gl_FragData[3] = discardFrag;
  }
}
```

B.3 Vertex Shader 2

```
void main()
{
  gl_TexCoord[0] = gl_MultiTexCoord0;
  gl_Position = ftransform();
}
```

B.4 Fragment Shader 2

```
uniform sampler2D texture1,texture2,texture3,texture4;
uniform int textureToRender;
uniform int no_of_horizontal_slices;
uniform float slice_resolution;
uniform float half_number_of_slices;
uniform float no_slices_per_texture;
uniform float view;
// View 0 is left, increasing this number increses view
// to the right, 1.0 should be the right most view within
// the eye separation.

// The following variables are calculated once in the main
```

```
// program to save on unecassary repeated computation.
uniform float pixel_width;
uniform float offset;
uniform float pixelwidth_times_view;
uniform float divide_By_NoOfHorizontalSlices;

void main()
{
  vec4 totalFrag = vec4(0.0,0.0,0.0,0.0);
  vec4 texel;
  float texelAlpha;
  vec3 texelColour;
  float minAplhaClamp = 0.20;

  float sliceNo = 0.0;
  vec2 texCoord;
  float base_position;

  // only the top left slice is visible in the program
  // therefore move the texture coordinates to this area.

  texCoord.st = gl_TexCoord[0].st;

  float no_slices_per_texture_times_two =
    no_slices_per_texture * 2.0;
  float no_slices_per_texture_times_three =
    no_slices_per_texture * 3.0;

  // composite all the slices in the 1st texture unit
  for(int y = 0; y < no_of_horizontal_slices; ++y) // columns
  {
    texCoord[1] = gl_TexCoord[0].t -
      float(y)*divide_By_NoOfHorizontalSlices;
```

```
for(int x = 0; x < no_of_horizontal_slices; ++x) // rows
{
  base_position = gl_TexCoord[0].s +
    float(x) * divide_By_NoOfHorizontalSlices + offset;

  // texture unit 0
  texCoord[0] = base_position - sliceNo *
    pixelwidth_times_view;
  texel = texture2D(texture1,texCoord.st);
  totalFrag = totalFrag + texel;

  // texture unit 1
  texCoord[0] = base_position -
    (sliceNo + no_slices_per_texture) *
    pixelwidth_times_view;

  texel = texture2D(texture2,texCoord.st);
  totalFrag = totalFrag + texel;

  // texture unit 2
  texCoord[0] = base_position -
    (sliceNo + no_slices_per_texture_times_two) *
    pixelwidth_times_view;

  texel = texture2D(texture3,texCoord.st);
  totalFrag = totalFrag + texel;

  // texture unit 3
  texCoord[0] = base_position -
    (sliceNo + no_slices_per_texture_times_three) *
    pixelwidth_times_view;
```

```
        texel = texture2D(texture4,texCoord.st);
        totalFrag = totalFrag + texel;

        sliceNo+=1.0;
      }
    }

    totalFrag.a = (totalFrag.a / (totalFrag.a + 1.0)) * 0.6;
    if(totalFrag.a < minAplhaClamp)
    {
      totalFrag.a = minAplhaClamp;
    }

    totalFrag.r = totalFrag.r * totalFrag.a;
    totalFrag.g = totalFrag.g * totalFrag.a;
    totalFrag.b = totalFrag.b * totalFrag.a;

    totalFrag.a = 1.0;
    gl_FragColor = totalFrag;
}
```