Technical Report

UCAM-CL-TR-382
ISSN 1476-2986

Number 382

UNIVERSITY OF CAMBRIDGE

Computer Laboratory

Synthetic image generation for a multiple-view autostereo display

Oliver M. Castle

October 1995

15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom
phone +44 1223 763500

https://www.cl.cam.ac.uk/

This technical report is based on a dissertation submitted April 1995 by the author for the degree of Doctor of Philosophy to the University of Cambridge, Wolfson College.

Technical reports published by the University of Cambridge Computer Laboratory are freely available via the Internet:

https://www.cl.cam.ac.uk/techreports/

ISSN 1476-2986
DOI *https://doi.org/10.48456/tr-382*

Abstract

Multiple-view autostereo displays provide true 3D images which offer depth cues (such as binocular stereo and motion parallax) that are not available with conventional monoscopic displays. This thesis addresses some of the problems of synthetic image generation for a multi-view autostereo display, with particular attention paid to the question of how rendering costs may be reduced by taking advantage of the similarities between the views.

A functional description of the prototype multi-view autostereo display device developed separately at the University of Cambridge sets the technological background of this research. The problems faced by synthetic image generation in general are reviewed next, of which visible surface determination is identified as the most important for multi-view stereo. A viewing model for multi-view stereo is then derived, building on experience with existing monoscopic and two-view stereoscopic viewing models.

Using this multi-view autostereo viewing model as a framework, two distinct approaches to multi-view stereo image synthesis are investigated. The first is an extension of conventional Z-buffer rendering methods, adapted to take advantage of the coherence between the views to eliminate redundant processing and share computational overheads wherever possible. The second, based on approximate stereo reprojection techniques, shares visible surface information between the views in an attempt to eliminate processing those parts of the scene considered unlikely to be visible in the final image, thus trading off image quality against rendering speed.

An experimental evaluation of these two techniques demonstrates that both are capable of producing multi-view stereo images at a lower cost per view than similar single-view rendering methods. The results indicate that the performance improvements of both algorithms increase with the number of views in the image, reaching asymptotic levels as the shared processing costs become relatively less significant compared with the overall rendering time. Of the two however, the approximate algorithm appears to offer the better potential speedup, owing to the way in which it enables the effective depth complexity of the scene to be reduced.

Preface

Except where stated, this dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration.

This dissertation is not substantially the same as any that has been or is being concurrently submitted for a degree or diploma or other qualification at any other University.

Unless otherwise stated, all the software used in experiments described herein is the original work of the author.

Acknowledgements

The research described in this dissertation has been carried out in the Computer Laboratory by the kind permission of Professor Roger Needham, Head of Department.

I would like to thank Neil Wiseman for the guidance and encouragement he has given me as my supervisor in the Computer Laboratory. His quiet reassuring manner and thoughtful advice have been invaluable. I would also like to thank Stewart Lang as leader of the Autostereo Display project in the Computer Laboratory for his support and interest in my work, and Adrian Travis as the inventor of the Autostereo Display.

There are many members (past and present) of the Rainbow Group who have helped make my stay in the Computer Laboratory stimulating and enjoyable, including: Jeremy Ball, Neil Dodgson, Chris Faigle, Jane Hunter, Rynson Lau, Olivia Nagioff, Uwe Nimscheck, Jon Sewell, Nicko van Someren, Martin Turner, and Adrian Wrigley, as well as honourary Rainbow members Margaret Levitt, John Moore, and Eileen Murray. I would particularly like to thank Neil, Chris, Uwe, and Jon for their helpful comments about earlier drafts of this dissertation. Thanks also to Jon Clarke, Heidi Day, and Steven Hird for being such a great bunch of people to share a house with.

I am grateful for the financial support of the Cambridge Commonwealth Trust Prince of Wales Scholarship, British Telecom Scholarship, New Zealand University Grants Committee, and the CVCP Overseas Research Students Awards Scheme during my research.

Finally, I would like to thank my parents, Trevor and Carolyn, to whom this thesis is dedicated, for their continuous encouragement, support, and guidance.

Contents

List of Figures

List of Tables

1

Introduction

In this chapter, the fundamental concepts of three-dimensional visual perception and synthetic image generation are introduced. True three-dimensional (3D) display, which exploits the power of human binocular vision, is distinguished from conventional 3D display, which relies on monocular cues to provide depth information in an image. Existing true 3D display technologies are briefly reviewed, and the opportunities and challenges presented by true 3D are outlined.

1.1 Three-dimensional depth cues

Most conventional computer graphics images which are considered "3D" are in fact merely 2D projections of 3D subjects: essentially flat views without inherent depth (Sexton [1989]). Information about the "missing" third dimension — depth — has conventionally been represented by the use of a variety of visual cues which are intended to illustrate how the appearance of entities changes with their depth from the observer. Some of the depth cues most commonly mentioned in the literature (McAllister [1993]) include:

linear perspective is a direct product of the perspective projection of 3D space onto a 2D plane. It causes the apparent size of an object to vary in inverse proportion to its distance from the observer: the further away an object, the smaller it appears. For correct interpretation of linear perspective cues however, the

observer's own past experience and knowledge of the size of objects in the world often plays an important part[1].

interposition (also known as *occlusion*) is derived from a property of opaque objects which overlap other objects along the same line-of-sight: nearer objects hide more distant ones. Because of the way in which this occurs, interposition is a very strong depth cue.

lighting and shadows can supply cues as to the relative positions of objects in view by the way in which they are illuminated by the light sources (typically from above) and how they cast shadows on other objects. This may however require considerable perceptual processing by the observer, and is thus not as strong a depth cue as some.

texture gradient can help provide additional clues about how far away an object is by the amount of detail visible on it: more distant objects generally exhibit less detail than nearer ones. However what may be considered to constitute detail is largely a matter of apparent scale which is influenced by perspective effects, and thus there is an element of learning associated with this as a guide to depth.

aerial perspective is perhaps the most common of a number of visual artifacts which may be attributed to the nature of the atmosphere of our planet: more distant elements tend to appear less distinct and cloudy, a phenomenon that is readily observable (in the extreme) on misty or foggy days. Other related effects include the appearance of mirages in the distance, or the colour and apparent size of the sun and moon near the horizon. Because of the large distances often involved however, these are typically relatively weak depth cues.

The depth cues listed above can be characterised as *psychological* in nature (McAllister [1992a]), in the sense that they are a by-product of the observer's personal experience about how the visual appearance of the world relates to its physical three-dimensional reality. However, there are other depth cues which are more closely related to inherent aspects of human *physiology* in the way in which they work:

accommodation is the change in focal length of the eye's lens as it focusses on objects at different distances from the observer. Due to the finite (non-pinhole)

[1]For example, consider a scene depicting a mouse and an elephant, where the image of the mouse appears to be the same size as the image of the elephant. Using previously-learned knowledge about the relative sizes of typical mice and elephants, an observer might reasonably interpret this to mean that the mouse is much closer than the elephant, but this information is not explicitly represented in the image itself.

size of the iris in the human eye, there is a limit to how large a range of depth can be in focus at any one instant. This gives rise to a phenomenon known as *depth of field*, where objects appear increasingly out-of-focus the further they are from the focal distance. However, as the focal distance increases, the effect weakens considerably, allowing only relatively coarse depth discrimination.

binocular convergence (or simply *vergence*) refers to the way in which the lines of sight of the left and right eyes meet at a common point of interest. The amount the eyes are turned inwards depends on the distance of the point from the observer. Like the accommodation cue however, this effect is most pronounced for nearby objects, and tends to have little impact at long distances.

binocular disparity between the images formed on the retina of the left and right eyes is a natural by-product of the distance between the eyes. This difference allows the brain to determine the apparent distance of an object from the observer by triangulation. This is a very strong and compelling depth cue which has its greatest effect at close range.

motion parallax (or simply *parallax*) is produced when the observer can see different views of a scene by moving in relation to the image. Objects that are nearer the observer appear to move more than those further way when inspected in a natural "look-around" manner. This can be a powerful depth cue that is analogous in many ways to that provided by binocular disparity, and often helps to resolve ambiguities in complex scenes. A similar (but not identical) effect may be observed when the viewer remains stationary and objects in the scene move.

Of these depth cues, only two of the physiological cues — binocular convergence and binocular disparity — rely on separate views being delivered to the left and right eyes independently, and thus require true 3D display techniques. The other depth cues are essentially monocular in nature, and may be used even without true 3D display. This is what much of conventional "3D" synthetic image generation has traditionally concentrated its efforts on: the reproduction of a simulated 3D environment as projected onto a flat 2D image plane.

1.2 True 3D display

True three-dimensional display of computer graphics images provides the observer with valuable additional information about the spatial relationships between objects represented in the image, in a way which conventional display techniques cannot. The essence of true 3D display is to present a different image to the observer's

left and right eyes independently, each image corresponding to what would be seen from the viewpoint of each eye. This permits the observer to use his natural powers of binocular depth perception to interpret the views seen by both eyes together as a single coherent 3D image (Brewster [1856]). This process, known as *stereopsis*, is illustrated in Figure 1.1.

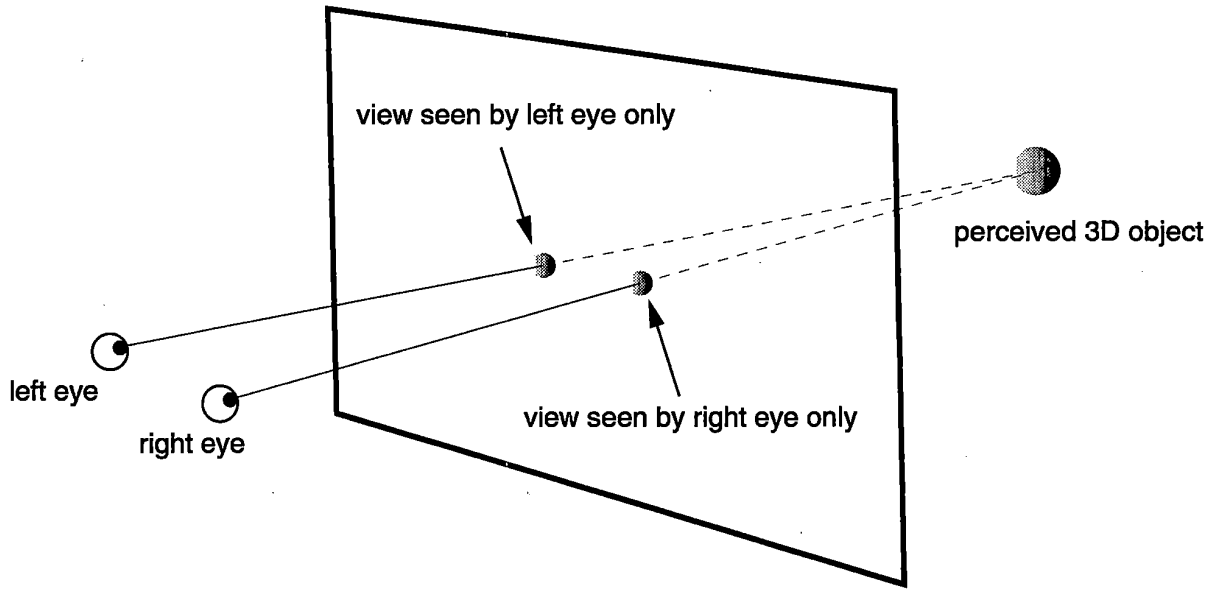


Figure 1.1: Stereopsis combines the view seen by each eye into a single true 3D image

Without true 3D display, both eyes see exactly the same view, and the image thus appears physically flat and two-dimensional. Only the subjective content of the image provides the monocular depth cues which allow the observer to interpret it as the intended 3D scene.

1.2.1 True 3D display technologies

Many different techniques have been proposed to display true 3D images, of which only a handful have been developed to a point where they can be used outside a research laboratory. A general review of common 3D display technologies may be found in McAllister [1993] or Lane [1982]. Further references to a wide variety of true 3D display technologies (including patent information) are listed in Starks [1991] and Starks [1992]. While it is not intended to replicate this information here, it is useful to present a brief summary and characterisation of the various techniques for true 3D display.

The fundamental issue for any true 3D display device is how the left and right eyes of the observer see a different view depending on their position with respect to the 3D scene. There are three basic approaches to this problem, which may be characterised as either *volumetric, holographic,* or *binocular stereoscopic.*

Volumetric displays (also known as *space filling* displays) portray a 3D scene directly in space. The view seen by each of the observer's eyes depend on their position with respect to the image, in the same manner as when the observer is looking at a real object. The scene is represented as a set of individually-illuminated points in a finite three-dimensional display volume. Typically, only a subset of these points can be illuminated simultaneously, so the device needs to repeatedly scan over the display volume at a rate fast enough to fool the human eye into perceiving a solid, continuous image. The resulting 3D image provides the full range of physiological depth cues, but cannot support all the psychological cues, including the important interposition cue. Its translucent point cloud image format is thus best suited to sparse 3D scenes with few (if any) solid surfaces. This places severe limits on its range of application.

Holographic displays represent a 3D scene indirectly by optically reconstructing the light wavefront reflected by each object in the scene. Each of the observer's eyes receives only the light reflected directly into the pupil, as occurs when looking at a real object. A holographic image may exhibit the full range of both psychological and physiological depth cues, and is generally considered to be the ideal 3D display medium. A *true hologram* exhibits continuous parallax in both the horizontal and vertical directions, and is commonly produced by recording the complex diffraction pattern of laser light reflected from a physical object. While it is possible to synthesise such a hologram digitally, the enormous amount of information represented in a true hologram makes it extremely expensive computationally. One way to alleviate this problem is to eliminate the vertical parallax in the image, while still retaining the horizontal parallax necessary to provide binocular and look-around depth cues — this is the *horizontal parallax only* (HPO) approach. Alternatively, the continuous parallax of a true hologram may be approximated using a relatively small number of discrete perspective views, as in a *holographic stereogram.* In addition to the benefits of bandwidth reduction that this allows, the other main advantage of this approach is that any set of conventional photographic or computer generated images which exhibit the appropriate parallax relationships can be used.

Binocular stereoscopic displays represent a 3D scene indirectly using a single pair of 2D views of the scene, one for each eye of the observer. A variety of different view selection techniques may be used to ensure that only the appropriate view is seen by each eye. Parallel selection uses either physically disjoint viewing channels, or optical filters such as colour anaglyphs and polarised images. Sequential selection uses mechanical or electro-optical shutters synchronised to a time-multiplexed display. While most binocular selection techniques require the observer to wear special eyeglasses of some kind, it is possible to avoid this with parallax barrier or lenticular lens arrays to produce an autostereoscopic display, although these impose certain constraints on the viewing position. A binocular stereoscopic image supports all the psychological depth cues but is deficient in the physiological cues, being generally unable to support dynamically varying accommodation and motion parallax effects. Despite this, binocular stereoscopic displays are capable of providing a usable true 3D image at a fraction of the cost of alternative approaches.

Of these three approaches, by far the single most common true 3D technique in use at the present time is binocular stereo using sequential selection. Arguably, this is attributable to the relatively small technology gap between this display format and most 2D displays available today. This degree of compatibility allows conventional 2D images to coexist with true 3D images on the same physical display screen, which makes true 3D an optional add-on rather than a replacement for current display technology. In the long term however, it may be expected that true 3D will become a standard feature, in the same way that colour superseded black and white displays (Faris [1994]). The demand for greater ease-of-use means that autostereoscopic displays are more likely to succeed in a mass market environment than binocular stereo devices which require the observer to wear special glasses of any kind (Lipscomb [1989]). Extending this ease-of-use principle further, it is suggested that multiple-view autostereoscopic displays, which impose fewer restrictions on viewer position, are even more likely to succeed. Multi-view autostereo technology provides the motivation for the synthetic image generation algorithms presented in this thesis, although many of the principles involved apply equally well to the perspective source views used in holographic stereograms.

1.2.2 The challenge of true 3D display

Apart from the physical engineering problems associated with true 3D display devices, the main challenge facing true 3D lies in the greatly increased amount of information necessary to describe a true 3D image in comparison with a conventional 2D image. While even the simplest binocular stereo image requires twice the raw

bandwidth of a 2D image of the same resolution, multiple-view autostereo images and holographic stereograms demand bandwidth increases in direct proportion to the number of views they contain. Holograms of comparable resolutions require bandwidths several orders of magnitude greater still.

From a computer graphics standpoint, ways need to be found to reduce the computational effort to synthesise true 3D images to manageable proportions. The ultimate goal is to find techniques for rendering true 3D images in the most efficient manner possible. This involves looking for and eliminating redundant information in the image, taking advantage of the spatial coherence of the 3D scene.

1.3 Outline

Having established the background of true 3D display techniques in general, Chapter 2 describes the particular multi-view autostereo display device used in this research, as well as the computer graphics system used to support it. An overview of synthetic image generation techniques in general is given in Chapter 3, providing the background for the work described in following chapters. This leads onto the development of the viewing model for multi-view stereo images in Chapter 4, which provides the basis for many of the techniques developed later.

Two new image synthesis algorithms for multi-view stereo are presented in Chapters 5 and 6. Chapter 5 describes in detail a number of ways in which the similarity between the views in a multi-view stereo image may be taken advantage of to improve the efficiency of conventional rendering techniques applied to stereo images. An alternative approach to image synthesis for multi-view stereo based on approximate rendering methods is given in Chapter 6. These new techniques are evaluated and compared in Chapter 7, both in terms of experimentally observed objective performance and subjective image quality. Finally, the results of this work are summarised in Chapter 8.

2

The Cambridge Autostereo Display

The *Cambridge Autostereo Display* is a multi-view autostereo 3D display device developed at the University of Cambridge (Travis [1990]; Travis and Lang [1991]). It has two main advantages over most alternative stereo displays (Lang et al. [1992]). The first is that no special viewing equipment needs to be worn by the observer in order to see the 3D effect and the image may be seen by the naked eye under normal room lighting conditions. The second is that the display's multiple viewpoints provide a "look-around" capability which enables the observer to see different 3D views of the image depending on his viewing position. Existing prototypes of the display support a number of different autostereo image formats, with up to sixteen monochrome or six colour stereo views in resolutions varying from 320×240 pixels up to 640×480 pixels.

This chapter describes the technological background against which the research into novel synthetic image generation techniques is set. First the fundamental principles of how the display operates are examined and a selection of the most important design issues for the display are discussed. The specific architecture of the display used in this research is then described in some detail, with particular reference to the display interface and the nature of the input signals required to produce a 3D image on the display. Finally, an outline of the experimental computer graphics system developed to drive the display is presented.

2.1 The principle of operation

The fundamental principle behind the operation of the display is the same as many other stereoscopic 3D displays (Travis [1990]). That is, a 3D image of a given scene can be perceived by an observer if his left and right eyes each see a different 2D view of the scene, corresponding to the difference between the positions of each eye. The display takes this principle one step further by producing multiple (more than two) views, where each view is visible only in a particular direction in front of the display. This provides an important instant "look-around" capability lacking in conventional 2-view stereo displays which makes it more natural to use. It also allows greater freedom of movement than is generally achievable with alternative autostereo technologies, without the need for tracking the position of the observer's head and the associated complications and latencies this introduces (Eichenlaub [1994]; Touris [1994]).

The key to how the display produces a multi-view autostereo image is in the combined application of two simple optical principles:

- controlling where each view is visible, and
- time-multiplexing multiple views to maintain a steady 3D image

In the description that follows, technical details are generally omitted for the sake of clarity. These details are discussed further in Section 2.2.

2.1.1 Controlling where each view is visible

Consider a conventional self-illuminating image source, such as a CRT. Light from each point on the surface of the CRT screen is scattered over a wide arc (see Figure 2.1). An observer anywhere inside that arc can see the image on the CRT screen.

Now consider what happens if an opaque barrier with a slit cut in it is placed between the observer and the screen, as in Figure 2.2. Light from the image on the CRT can only be seen by an observer in the much narrower arc made visible by the slit.

By moving the position of the slit, the region in which the image on the CRT is visible can be controlled, as illustrated in Figure 2.3. Thus the slit acts as a kind of directional shutter, the position of which controls where the image on the CRT is visible from.

2.1.2 Time-multiplexing of views

There is a property of the human visual system, known as *persistence of vision*, which makes the eye insensitive to flicker in an image that is flashing on and off when

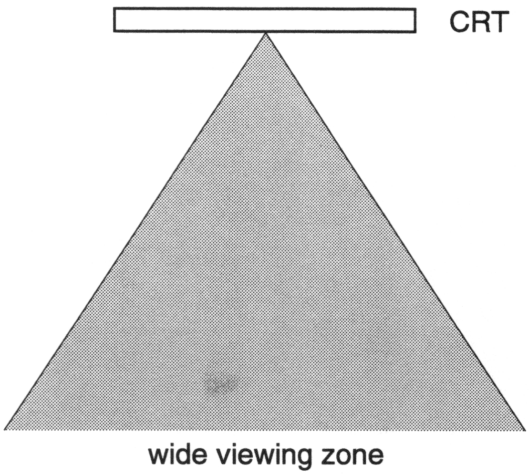


Figure 2.1: Region illuminated by CRT

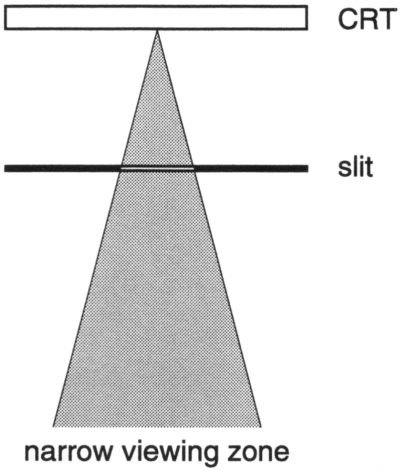


Figure 2.2: Region illuminated by CRT with a slit in front of it

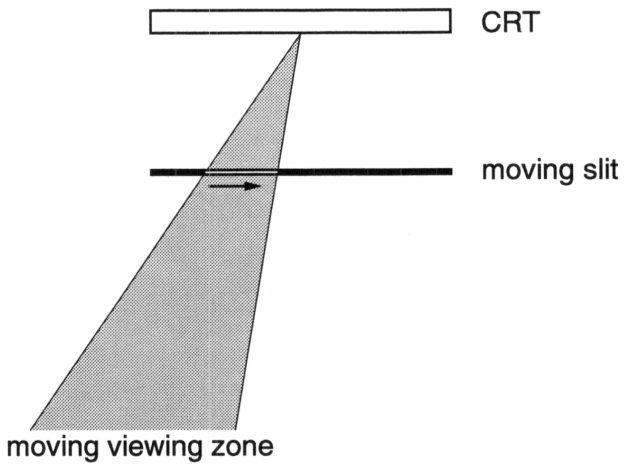
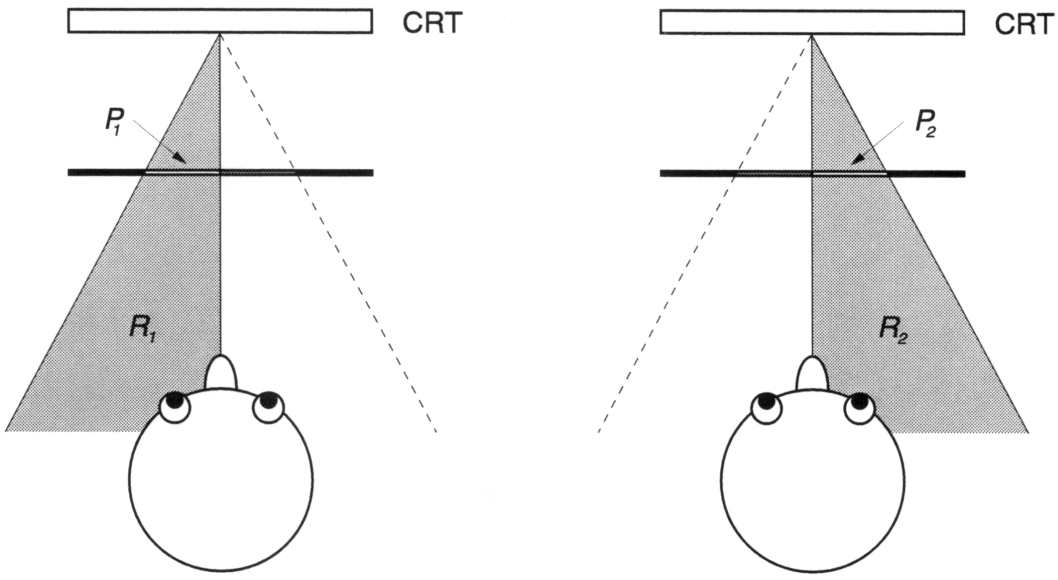


Figure 2.3: Region illuminated by CRT with a moving slit in front

the flicker frequency exceeds a particular rate, known as the *critical fusion frequency* (Davson [1980], chapter 12). This means that even when the image is only displayed for a fraction of a second at a time, as long as the rate of display of the image (known as the *refresh rate*) is sufficiently high, the eye will perceive a steady image without flicker. This is, of course, precisely the principle used by all cinemas and televisions. By combining this property of the human visual system with the control of where an image is visible from described the Section 2.1.1, it is possible to produce a flicker-free true-3D image, without encumbering the observer with any special viewing equipment.

Consider again the CRT with a movable shutter between it and the observer controlling where the CRT image is visible from, as shown in Figure 2.4. When the shutter is at position P_1, the image is visible in region R_1. Similarly, when the shutter is at position P_2, the image is visible in region R_2. With the observer's head positioned such that his left eye is in region R_1 and his right eye in region R_2, the left eye would see the image displayed when the shutter was in position P_1, and the right eye would see the image displayed when the shutter was in position P_2. This is illustrated in Figure 2.4(a) and (b) respectively. The left-eye view of a stereo pair is displayed on the CRT when the shutter is in position P_1, and the right-eye view of a stereo pair is displayed when the shutter is in position P_2, thus allowing the observer to see a stereo 3D image. By alternating the combination of shutter positions and left- and right-eye view images rapidly enough, the time difference between the display of the left- and right-eye views becomes imperceptible to the human eye and the stereo image appears flicker-free.

Now consider a similar set-up to that described above, but where there are more



(a) shutter position P_1 open (b) shutter position P_2 open

Figure 2.4: A model of a two-view autostereo display using a two-position slit

than two shutter positions with their corresponding images (see Figure 2.5). In this configuration, a number of different views are made visible in a number of discrete angular regions. As in the two-view case, a different view is displayed on the CRT for each shutter position at a rate sufficient for all views in the image to appear flicker-free. Assuming a sufficiently small separation between viewing regions[1], an observer can see different views in his left and right eyes (and thus perceive a stereoscopic 3D image) regardless of the position of his head in front of the display. Furthermore, as the observer's head moves from side to side the view seen by each eye changes, corresponding to the view which should be seen from that direction in relation to the display screen, while still maintaining stereo viewing. In this way, a multi-view autostereo image is produced, providing free-viewing true 3D with look-around.

The main problem with this approach is the comparatively high frame rate required by the display to make the multi-view stereo image appear flicker free. In general, an n-view time multiplexed stereo display requires a frame rate up to n times greater than that required for a conventional (single-view) display to eliminate apparent flicker in the image.

Another potential problem with this approach to stereo image display is the discrepancy between the time each view is displayed and the time of the event it is

[1] see Section 4.3.5

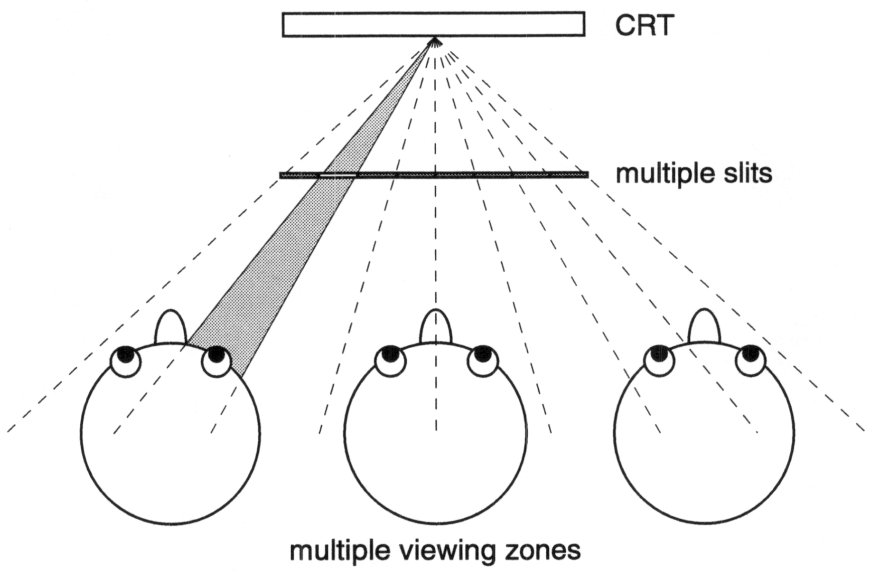


Figure 2.5: A model of a multi-view autostereo display using a multiple-position slit

intended to represent. Temporal artifacts of this kind may produce unwanted stereo aliasing effects with moving images. An example of such an artifact is the induced depth shift observed with a horizontally moving cursor in a two-view time-multiplexed stereo image, as reported by Butts and McAllister [1988]. This may occur when the observer's short term visual memory associates the view of a moving object seen in its position at time t_1 by one eye with another view of the same moving object seen in a different position at time t_2 by the other eye, resulting in the observer perceiving the wrong stereo disparity between the two views and thus a shift in the apparent depth of the moving object. The direction of the apparent depth shift depends on the order in which the views are displayed and the direction in which the object is moving. When the object comes to rest, the perceived disparity stabilises and the intended depth is correctly observed.

A solution to this problem is likely to be difficult to find in general, particularly as the apparent position of a given moving element in each view of the image ideally should depend on the precise timing of its display in each view. However this may not prove as troublesome with a multi-view autostereo display for two reasons. Firstly, the very high refresh rates demanded by such a display mean that the time between observing left and right eye views of any given image element is correspondingly less than in the two-view case. Secondly, the relative time difference between the display of the views seen by the observer's left and right eyes showing the position of the object at time t_1 is typically much less than the relative time

difference between the display of the view seen by the right eye showing the object at time t_1 and the display of the view seen by the left eye showing the object at time t_2 (assuming a simple left-to-right view display order), and thus the chance of the observer stereoscopically associating views from different time frames is less with a multi-view display than with a two-view display, where the differences between the left-right and right-left display times are equal.

2.2 Display design issues

In an autostereo display of this kind, there are two dominant practical considerations which define and constrain the design:

1. the need for many closely-spaced views to produce a high-quality autostereo image

2. the need to refresh each view at a fast enough rate to eliminate flicker

The views must be sufficiently closely-spaced in order to guarantee that each of the observer's eyes will see a different view at the expected viewing distance. However a higher quality look-around effect can be obtained by further reducing the spacing between views, thus reducing the differences from one view to the next and producing a more seamless multi-view image. Of course, the more closely-spaced the views, the greater the number of views required to cover a given overall field of view in front of the display.

There is a trade off between the number of views displayed and the refresh rate of each view. As the number of views increases, the time available to refresh each view (within a given flicker period) decreases, and thus faster refresh times are necessary for each view. Conversely, a given number of views in the image places implicit limits on either the quality of the look-around effect or the overall field of view of the displayed image. Furthermore, as the number of views increases, so the proportion of time that each view is displayed for decreases. Each view therefore needs to be displayed at a correspondingly greater brightness in order for an equivalent intensity to be perceived by the human eye[2]. Thus not only does the CRT used for the autostereo display need to be very fast, it also needs to be capable of very high intensity output.

The time required to refresh each view depends on the resolution of (number of pixels in) the view and the speed of the display device. In practice, there is a limit

[2]this is explained by the *Talbot-Plateau* law, which states that the perceived intensity of an intermittently-illuminated (but non-flickering) image is equal to the average brightness over the time interval between successive illuminations (Davson [1980], chapter 12).

to the number of pixels that can be output in a given time using a particular display device. Therefore, it may be possible to reduce the resolution of each view in order to display more views within a given time period. Conversely, if greater resolution is required, this may be obtained by displaying fewer views.

There is a limit to how much the time taken to display each view may be reduced however, determined by the persistence of the phosphor on the face of the CRT. If the time between display of successive views is too brief, then the phosphor glow from one view may not have had time to decay sufficiently before the next is displayed. This residual image would then appear to be visible from the direction corresponding to the next shutter position in addition to the direction intended, albeit at reduced intensity. An observer would see this as a fainter "ghost image" of the previously displayed view, superimposed on the correct view. This results in the views appearing to "bleed through" from one to the next in a manner which can significantly degrade the subjective quality of the autostereo image. A similar ghosting or channel crosstalk artifact can also arise if the contrast ratio of the shutter is too low[3], although this cannot be alleviated by altering the display timing. However, it should be pointed out that these problems are by no means unique to this particular display device — similar artifacts have been reported with other CRT-based time-multiplexed stereo displays (Lipton [1987]).

Another related practical problem concerns the time required to change the position of the shutter. The autostereo display relies on refreshing a each view only when its corresponding slit is open. Therefore it must be possible to change shutter positions in the time interval between the end of one view refresh and the start of the next. The switching speed of the shutter hardware thus determines an upper limit on the frame rate of the display, independent of the characteristics of the CRT unit.

2.3 The display architecture

There are two different versions of the display currently in use: one providing only monochrome output, the other providing colour. Both however share a good deal in common, including the essential mechanism for controlling the directional shutter. Most of the description which follows in Section 2.3.1 concerning the architecture of the monochrome display also applies to the colour version, with those features unique to the colour display described in Section 2.3.2.

[3]the contrast ratio of a shutter is defined as the ratio of the light transmission of the shutter in its open state to its transmission in the closed state

2.3.1 Monochrome display

The monochrome autostereo display consists of three major subsystems, as shown in Figure 2.6:

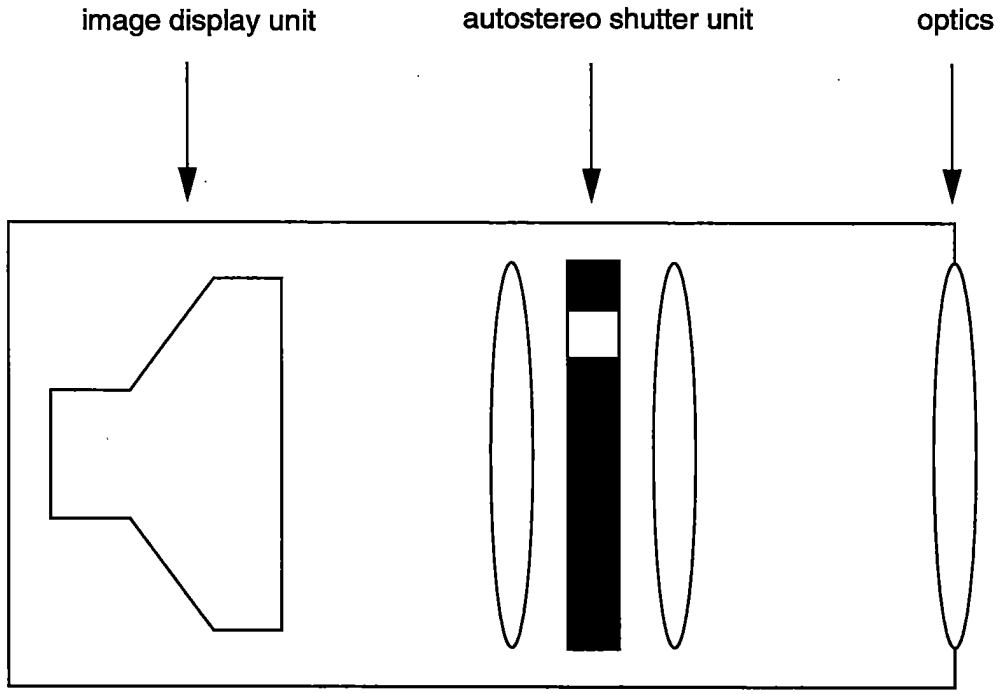


Figure 2.6: Block diagram of the monochrome autostereo display

image display unit a high-performance CRT, utilising a high-brightness, fast decaying white phosphor[4] tuned for high-frequency operation. It is capable of supporting video line rates of up to 150 kHz.

autostereo shutter unit a high-performance ferroelectric liquid crystal device (LCD). It has sixteen independently-controlled vertical slits, each of which can be switched between opaque and transparent states in less than 200 μs.

additional optical elements used to correctly focus the autostereo image on the screen at the front of the display. These are not relevant to this discussion: the interested reader is referred to Travis and Lang [1991] for details.

[4]so-called "TV white"

The image display unit and the autostereo shutter together place limitations on:

1. the number of views available in the autostereo image, and
2. the resolution of each view.

The maximum number of views that can be produced by the display is limited by the number of independently-controllable slits in the shutter, the switching speed of the shutter, and the refresh rate of the CRT. The current sixteen-slit LCD shutter can support an autostereo image with up to sixteen views. The number of views that can be displayed is not directly related to the switching speed of the shutter, but there must be sufficient time for the shutter to switch in the vertical flyback interval between CRT frame refresh cycles. Assuming that the vertical flyback takes approximately 20% of the total frame refresh time, then the 200 μs switching time of the current LCD implies a minimum CRT frame refresh time of 1000 μs, corresponding to a maximum frame refresh rate of 1000 Hz.

The resolution of each view is dependent on the video bandwidth of the CRT and the number of views in the image. The CRT video bandwidth is shared equally among all the views, with each individual view requiring a refresh rate above the flicker threshold. In addition, allowance must be made for the vertical flyback interval between frame refresh cycles which may account for up to 20% of the total time for each frame. Assuming a minimum acceptable flicker-free refresh rate of 50 Hz for each view, the 150 kHz line rate of the CRT is theoretically capable of supporting up to 300 lines per view in an eight-view autostereo image, or 150 lines per view in a sixteen-view image.

However, by exploiting the ability of the CRT to display video in a 2:1 line interlaced mode it is possible to halve the line bandwidth required to maintain a given vertical resolution and flicker rate. There is some penalty in terms of image quality, most noticeably in the appearance of flicker on single-pixel horizontal line segments, but this may be considered an acceptable trade-off for the effective doubling of apparent vertical resolution. Interlaced video is therefore used in all but one of the monochrome autostereo image formats currently supported, as listed in Table 2.1. The use of interlaced video with the autostereo display is examined further in Section 2.4.

From Table 2.1 it is possible to make several observations. Firstly, the perceived refresh rate of a given view is above the accepted minimum 50 Hz in all monochrome autostereo image display formats, with the highest perceived refresh rate belonging to the format with the fewest views (H6) and the lowest perceived refresh rate to the format with the greatest number of views (L16 and D16). Although it would be quite feasible to trade some of this apparent excess video bandwidth for modest increases in pixel resolution, in practice it is generally considered more

Table 2.1: Autostereo monochrome display formats

Format code	Views	Resolution (h×v pixels)	Actual CRT refresh rate (Hz)	Perceived refresh rate (Hz)	Pixel clock (MHz)
L8†	8	320×240	530	66	72
L16	16	320×240	960	56	72
D16	16	640×240	960	56	144
H6	6	640×480	530	75	144
H8	8	640×480	530	58	144

† non-interlaced

important to maintain a higher refresh rate in order to suppress flicker as much as possible. Secondly, the current display does not place as great a constraint on the horizontal resolution of the image as much as it does on the vertical resolution, as the pixel bandwidth of the CRT does not saturate as early as the line bandwidth. This can be seen by comparing formats L16 and D16, which differ only by a factor of two in the pixel clock and a factor of two in the horizontal resolution. In fact this is not an inconvenient property for the display to have, as the depth resolution of a stereo image is dependent on the horizontal pixel resolution alone. The depth resolution of a stereo image is discussed further in Section 3.3.3.

2.3.2 Colour display

The colour display is very similar to the monochrome version described in Section 2.3.1. It uses the same image display and autostereo shutter units as the monochrome version, but adds a third subsystem to handle colour: the colour shutter unit. See Figure 2.7.

The colour shutter unit applies a red, green, or blue filter to the otherwise monochrome white output of the image display unit. By sequentially displaying the red, green, and blue channels of each view in synchronisation with the appropriate colour shutter setting, a wide range of colour output can be produced. This sequential mixing of independent primary colours to produce a wider colour gamut relies on a similar mechanism of persistence of vision in the human visual system to that utilised in the display of flicker-free images.

The colour shutter unit used in the current colour autostereo display is a Tektronix NU700S nematic Pi-cell LCD with five independently-switchable rectangular segments. These segments are normally switched sequentially to follow the ver-

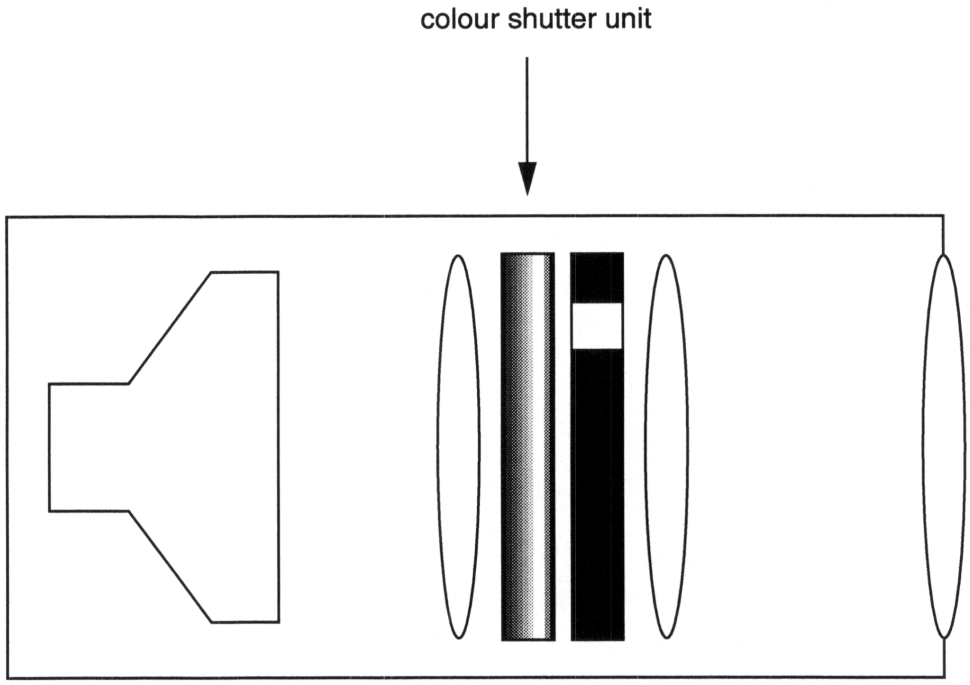


Figure 2.7: Block diagram of the colour autostereo display

tical scan of the electron beam down the face of a CRT, but for the autostereo display the shutter is rotated onto its side so that the segments follow the horizontal scan of the autostereo slit instead. It is capable of switching each segment in under 4 ms, allowing a maximum colour switching rate of 250 Hz.

In other respects, essentially the same fundamental hardware limitations described in Section 2.3.1 also apply to the colour autostereo display in terms of maximum number of views possible and maximum display speed. However the time sequential colour approach means that as the number of bits used per pixel increases from 8 for monochrome (greyscale intensity) to 24 for the colour display (8 for each of the red, green, and blue channels), so the video bandwidth required for an image of comparable resolution also increases by a factor of three. This increase in the demands on the video bandwidth of the CRT means that the effective maximum number of views that can be supported must be reduced to six instead of sixteen. The colour autostereo image formats currently supported (all of which use interlaced video) are shown in Table 2.2.

It can be seen from Table 2.2 that there is no opportunity with the current colour autostereo display to trade off the number of views against the vertical resolution, as done with the monochrome display described in Section 2.3.1. Nevertheless it is still possible to gain a useful increase in the horizontal resolution of the

Table 2.2: Autostereo colour display formats

Format code	Views	Resolution (h×v pixels)	Actual CRT refresh rate (Hz)	Perceived refresh rate (Hz)	Pixel clock (MHz)
L6C24	6	320×240	960	50	72
L6C8†	6	320×240	960	50	72
D6C24	6	640×240	960	50	144
D6C8‡	6	640×240	960	50	144

† this is an 8-bit source (3:3:2 RGB) colour emulation of L6C24
‡ this is an 8-bit source (3:3:2 RGB) colour emulation of D6C24

image by using a higher frequency pixel clock, as demonstrated by comparing formats D6C24 and D6C8 with L6C24 and L6C8. Note however that the perceived refresh rate is somewhat less than that of monochrome formats of similar resolutions in order to accommodate the time sequential colour format. The reason for this is that the video bandwidth requirements of a six-view autostereo image with three colour channels in sequential format is equivalent to that of an eighteen-view single-channel autostereo image at the same resolution, which is two views more than the maximum number of views used in any of the monochrome autostereo formats listed in Table 2.1.

2.4 The display interface

The interface to the autostereo display has much in common with conventional CRT-based displays. The time-multiplexed nature of the display's operation means that the video input signals for each view may all use the same physical connection. Only one additional signal (known as the ZSYNC signal) is required to allow the autostereo shutter position to be synchronised with the corresponding view on the video input. The format of the video input signals for both monochrome and colour displays is described in Section 2.4.1, while the autostereo view/shutter synchronisation signal is dealt with in Section 2.4.2.

2.4.1 Autostereo video input

Video input to the display is provided via two BNC connectors. The first supplies the intensity signal which describes the image pixel data itself, while the second provides a composite horizontal/vertical synchronisation signal. These sig-

nals are no different in general format to conventional RGB video signals, except that only one of the red, green, or blue channel intensity signals is actually utilised by the display. The principal difference between the autostereo video signal and conventional alternatives is the way in which the display interprets the signal. This mechanism is described in the following sections, first for the simple case of a non-interlaced video signal, and then for the more complex interlaced video case.

2.4.1.1 Monochrome autostereo video

On a conventional monoscopic CRT display, a series of successive non-interlaced frames appearing in the video input stream are interpreted as a series of images taken at successive points in time. In Figure 2.8, each rectangular box represents a single frame of non-interlaced video. Displaying each successive frame on the CRT as it arrives reproduces the same sequence of images obtained from the original image source for a single spatial view.

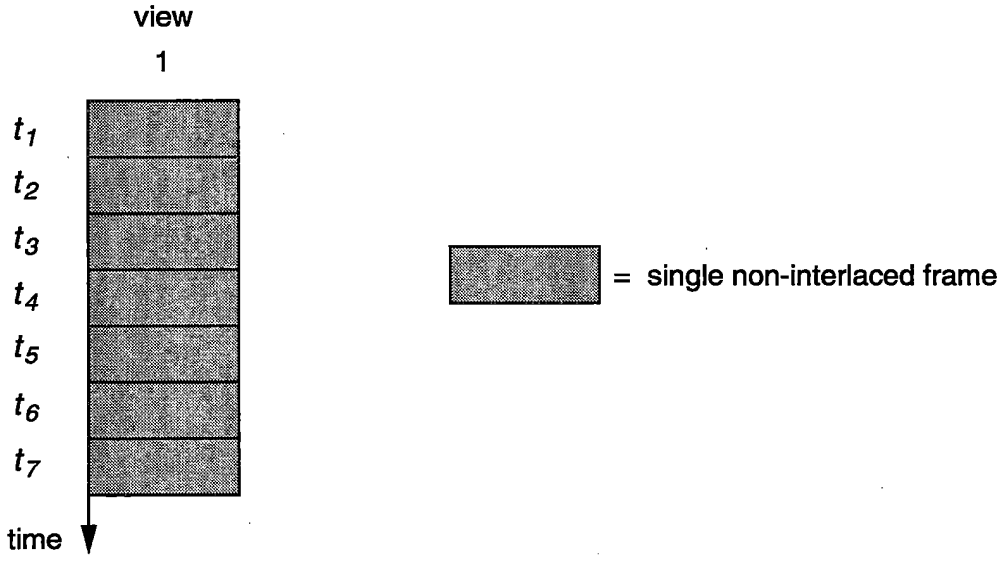


Figure 2.8: Format of conventional monoscopic non-interlaced video. A conventional CRT display interprets successive non-interlaced video frames as successive views in time

On the autostereo display however, a series of successive frames appearing in the video input stream is interpreted as a series of views taken from different points in space. Each view should be visible only from a particular direction, so the autostereo shutter position must be set accordingly as each frame is displayed. This is performed for each view and shutter position in the autostereo image, and then the

cycle is repeated for the views in the following autostereo image. In this way, the autostereo display reproduces both the spatial and temporal context of the views obtained from the source.

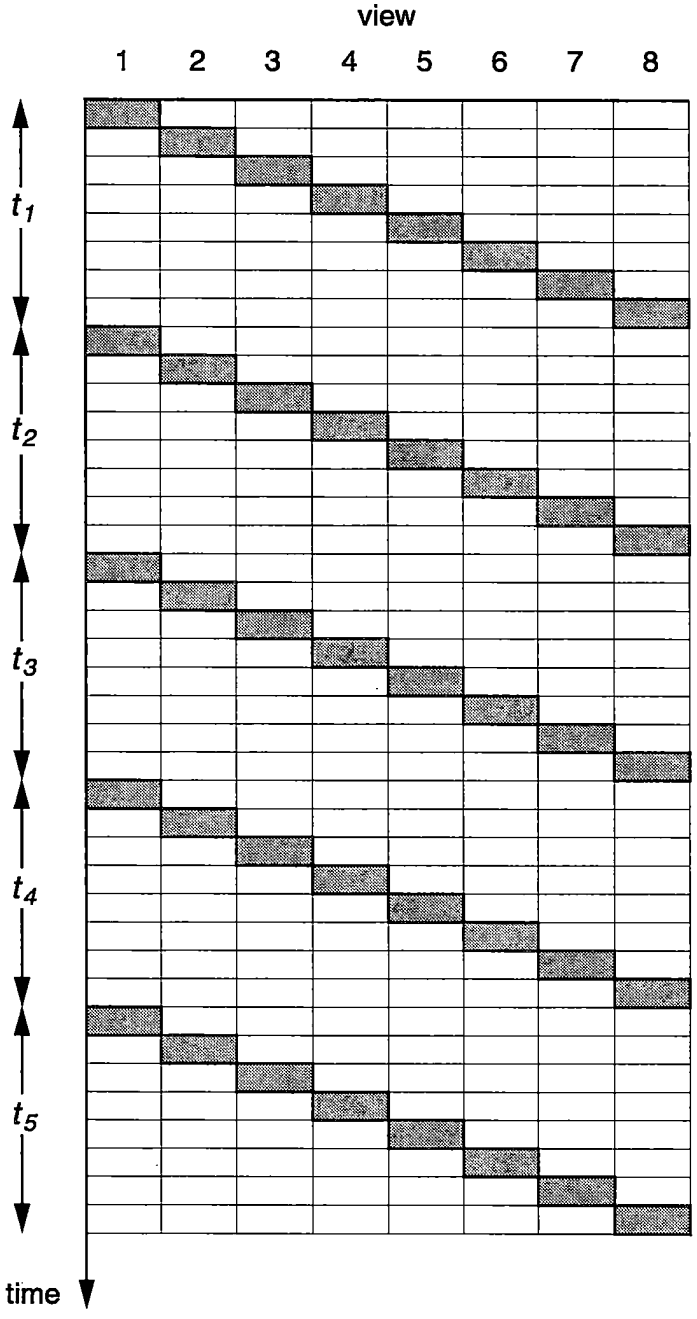


Figure 2.9: Format of autostereo non-interlaced video. The autostereo display interprets successive video frames as successive views in space as well as time.

This is illustrated for an eight-view image format (L8 from Table 2.1) in Figure 2.9. Each column represents a separate spatial viewing channel while each row down the page represents a successive frame time on the display. Each shaded box represents a frame displayed at a particular time in a given view. The set of frames which make up each autostereo image are shown grouped into successive time intervals $(t_1, t_2, \ldots)$.

As mentioned in Section 2.3, using interlaced instead of non-interlaced video effectively halves the bandwidth required to maintain an image of a given resolution at a specified refresh rate, while keeping flicker to acceptable levels. An interlaced video stream consists of a series of half-frames called *fields*. Each field is made up from half the scanlines normally found in a complete image. By alternating between displaying all the even numbered scanlines in one field and all the odd numbered scanlines in the next, an image of similar appearance to a single full resolution frame is produced (see Figure 2.10).

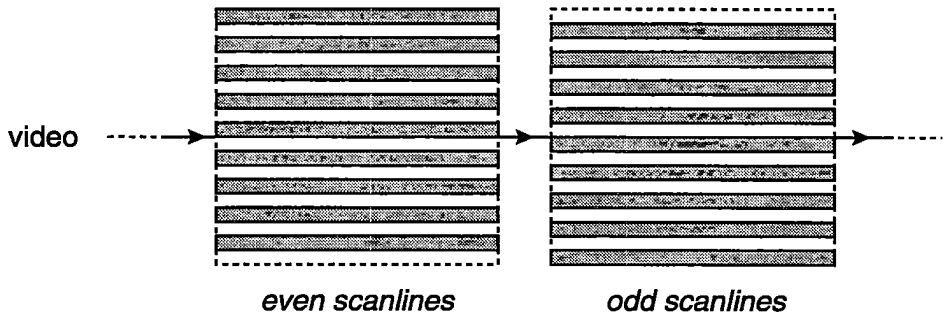


Figure 2.10: Interlaced video fields. Even and odd numbered scanlines are displayed on alternate successive interlaced fields.

On a conventional CRT, a series of successive interlaced video fields is displayed in much the same way as non-interlaced frames, that is as a series of images taken at successive points in time. This is illustrated in Figure 2.11. However there are two important differences between interlaced and non-interlaced display. The first is that the spacing between successive lines within a field is doubled. The second is that odd fields must be displayed at a one-line vertical offset with respect to the start of even fields, so that the scanlines in the even and odd fields maintain the correct relative positions.

On the autostereo display, a series of successive interlaced video fields is displayed in much the same way as non-interlaced frames: that is, as a series of views taken from different points in space. In order to avoid flicker, each view in the autostereo image must be refreshed at regular intervals. However matters are somewhat more complicated for an interlaced autostereo image than for a non-interlaced

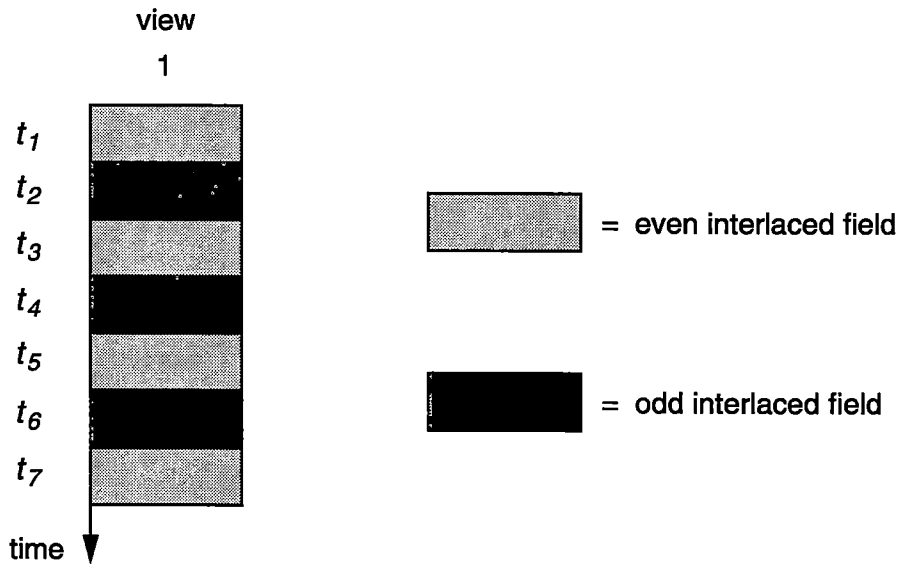


Figure 2.11: Format of conventional monoscopic interlaced video. A conventional CRT display interprets successive interlaced video fields as half-frame views taken at successive points in time.

one, as successive refreshes of a given view must alternate between the even and odd scanlines of that view in order for full vertical resolution to be achieved. It would be convenient if it were possible to display the even fields of all the views, followed by the odd fields of all the views, but standard interlaced video generation hardware does not support such a facility. Instead, even and odd scanlines are displayed on alternate successive fields, and thus there must be an odd number of fields separating the even and odd field refreshes of each view. For an interlaced autostereo image with an even number of views, it is therefore necessary to insert a dummy (blank) view into the video stream at an appropriate point. This is shown in Figure 2.12 for a six-view autostereo interlaced video stream (format H6 from Table 2.1).

2.4.1.2 Colour autostereo video

On a conventional CRT, multiple channels of colour information in the image are handled simultaneously in parallel. That is, the red, green, and blue colour components of each pixel are received at the same time and displayed by the CRT in parallel with each other. Depending on the type of inputs required by the CRT, separate physical cables may be used for the red, green, and blue colour components, or all colour channels may be combined in a single physical cable. In any case, the timing

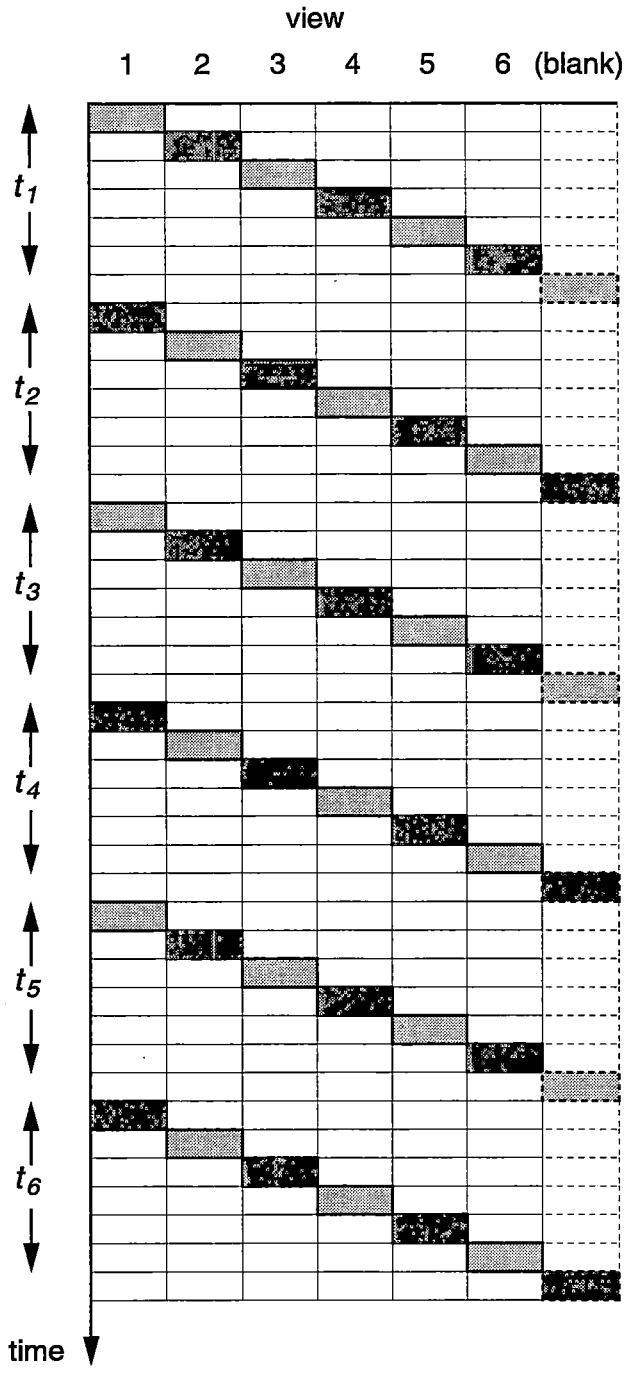


Figure 2.12: Format of autostereo interlaced video. The autostereo display interprets successive interlaced video fields as successive half-frame views in space and time. The dummy blank view is required for an interlaced autostereo image with an even number of views to ensure that even and odd fields are displayed on alternate refreshes of each view.

of the input signals used by a conventional CRT is essentially the same regardless of the number of colour channels in the signal.

In contrast, the autostereo display handles multiple colour channels in a time-multiplexed format, as described in Section 2.3.2. The red, green, and blue components of each pixel are received at different times on the same physical input channel and displayed by the CRT in time-sequential fashion. As each colour component is displayed, the colour shutter in the display is set to filter the appropriate hue. Due to the relatively slow switching speed of the colour shutter, it is necessary to display the individual channel intensities for all views before switching to the next colour channel. Thus the red channel of all the views are displayed first, followed by the green channel of all the views, and finally the blue channel of all the views. Figure 2.13 illustrates this for six-view RGB colour sequential autostereo (format L6C24 from Table 2.2). In all other respects, the handling of the different views and switching of the autostereo shutter is essentially the same as that described in Section 2.4.1.1 for monochrome video.

2.4.2 Autostereo view and shutter synchronisation

With a conventional (single-view) CRT, each frame in the video input stream represents a single complete static image. In contrast, with the autostereo display a single complete static image is made up of a number of frames in the video input stream. Each frame represents only a single view of the full autostereo image, and each view requires its own shutter position to ensure it is only visible from the appropriate direction. To synchronise the display of each frame with its corresponding shutter position, the display must know two additional pieces of information: how many frames make up the complete autostereo image, and which view the current frame represents. This is done using an additional synchronisation input to the display called the ZSYNC signal.

The ZSYNC signal is provided on a standard BNC input, and is a simple binary signal used to delimit the frames of each full autostereo image. A ZSYNC pulse is sent during the last frame of each autostereo image, as shown in Figure 2.14 for an eight-view monochrome image (format L8 from Table 2.1). By counting the number of frames from one ZSYNC pulse to the next, the display can determine the number of views in the autostereo image. As each subsequent frame arrives, the display sets the autostereo shutter position according to the expected number of views in the autostereo image and the number of frames which have elapsed since the previous ZSYNC pulse[5].

[5] this mechanism assumes that the next autostereo image will have the same number of views as the last. This is the case for a display operating in the same mode continuously, although mode changes can cause momentary disturbances on the display before synchronisation is regained.

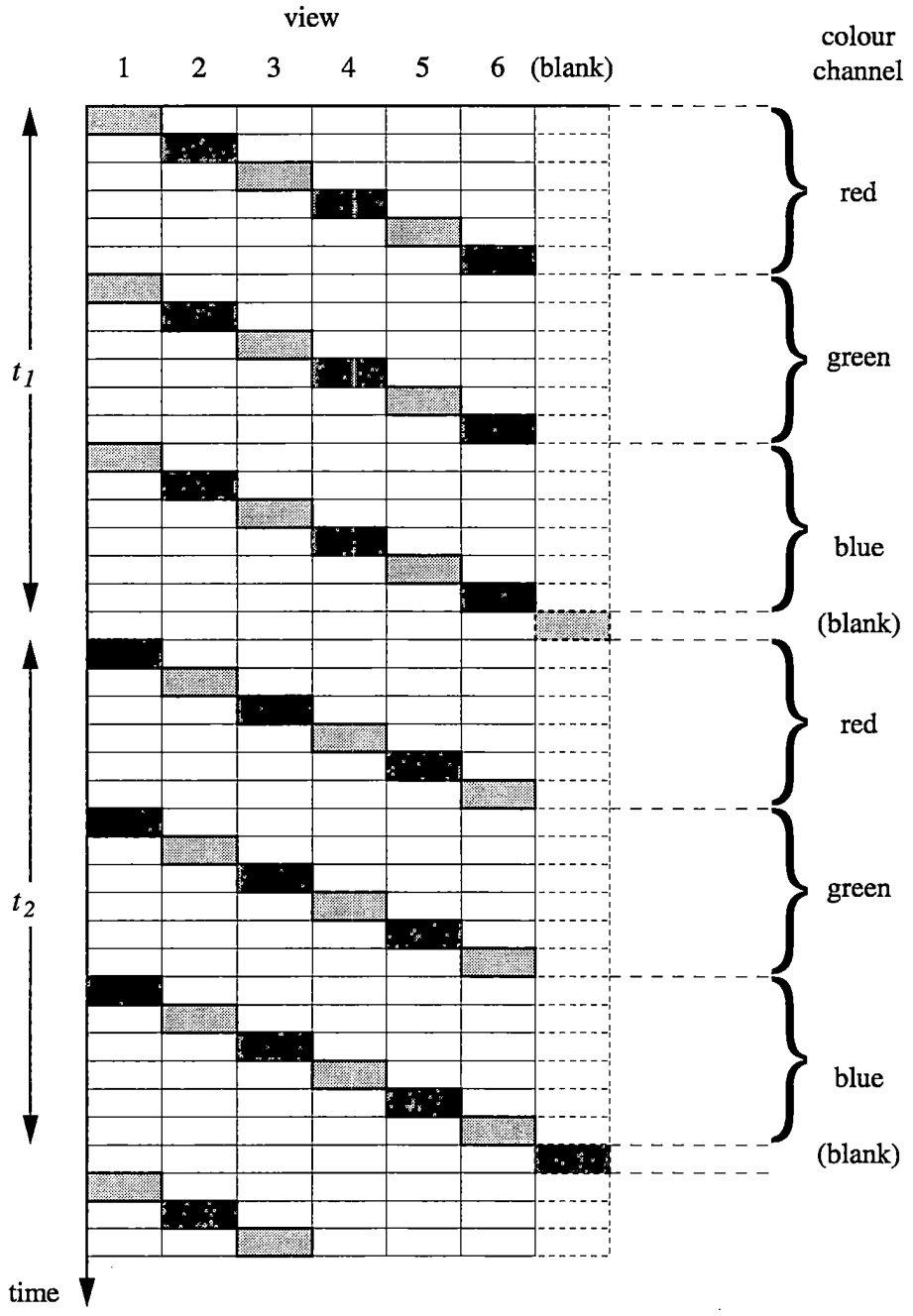


Figure 2.13: Autostereo multi-channel colour sequential video

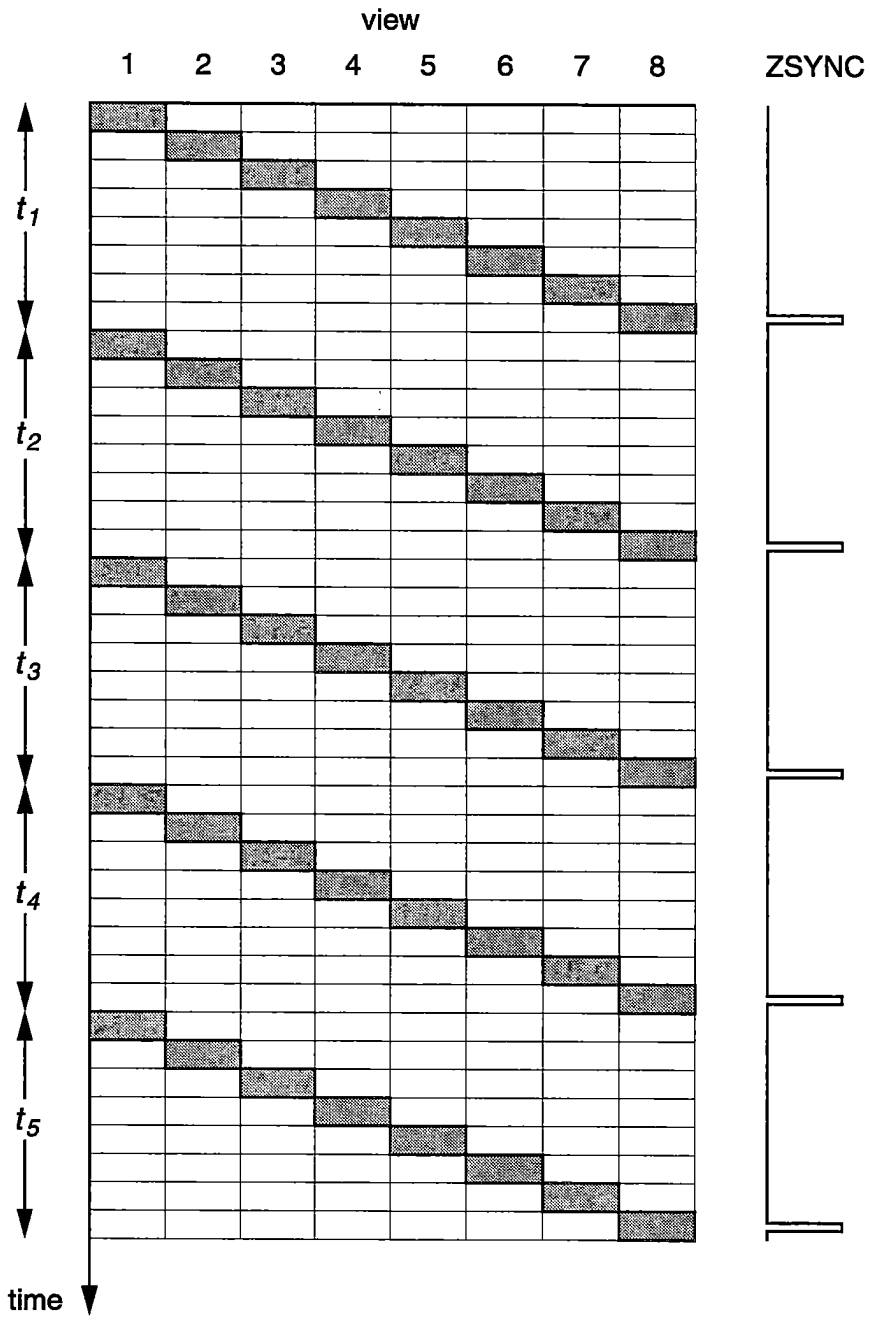


Figure 2.14: The relationship of the ZSYNC pulse to a sequence of autostereo video images

2.5 The computer graphics support platform

The autostereo display is capable of being driven from any source which can provide inputs of the format described in Section 2.4. For computer graphics purposes, the support platform selected must to satisfy all these fundamental requirements in addition to providing a means of specifying the images to be displayed. This section describes the hardware and software components of the support platform used for the autostereo display and how it meets these requirements. First the basic graphics hardware platform is described, followed by the low-level display device driver software which controls the video output to the autostereo display. The applications programming interface for this platform is then outlined, and a brief overview is presented of the environment in which application programs are developed and executed for the autostereo display.

2.5.1 The hardware platform

The essential video output requirements placed on the computer graphics hardware by the autostereo display are:

1. a high speed frame buffer large enough to store all views of the autostereo image, and

2. an external binary output for the ZSYNC signal.

The Merlin graphics board (Datapath [1991b]) is a readily available off-the-shelf card which meets these requirements. It has a 4 Mbyte frame buffer capable of supporting pixel output bandwidths of up to 225 MHz, and an on-board serial output port suitable for the ZSYNC signal.

In addition to these fundamental requirements, the Merlin has a number of other features which make it well suited to use with the autostereo display. It has a Texas Instruments TMS34010 Graphics System Processor (GSP) which allows very flexible programmable control of video output formats (Asal et al. [1986]; Texas Instruments [1988]). It also has an Intel i860 CPU, which is suitable for more general computing tasks including 3D rendering and image processing (Grimes et al. [1989]; Margulis [1990]). Both the TMS34010 and i860 processors share access to the Merlin's 4 Mbyte VRAM frame buffer as well as its 16 Mbyte DRAM for user programs and data.

The Merlin board has an ISA bus interface and is designed to be used in an IBM-compatible PC. In the configuration used in all the work carried out as a part of this thesis, it has been installed in a 33 MHz 486-based PC with 8 Mbyte of RAM and

a 110 Mbyte hard disc. In addition, the PC has an Ethernet connection to the local Computer Laboratory network and uses PC-NFS software to allow images and data to be accessed from a variety of sources. A simple functional block diagram illustrating the hardware configuration of the system is shown in Figure 2.15.

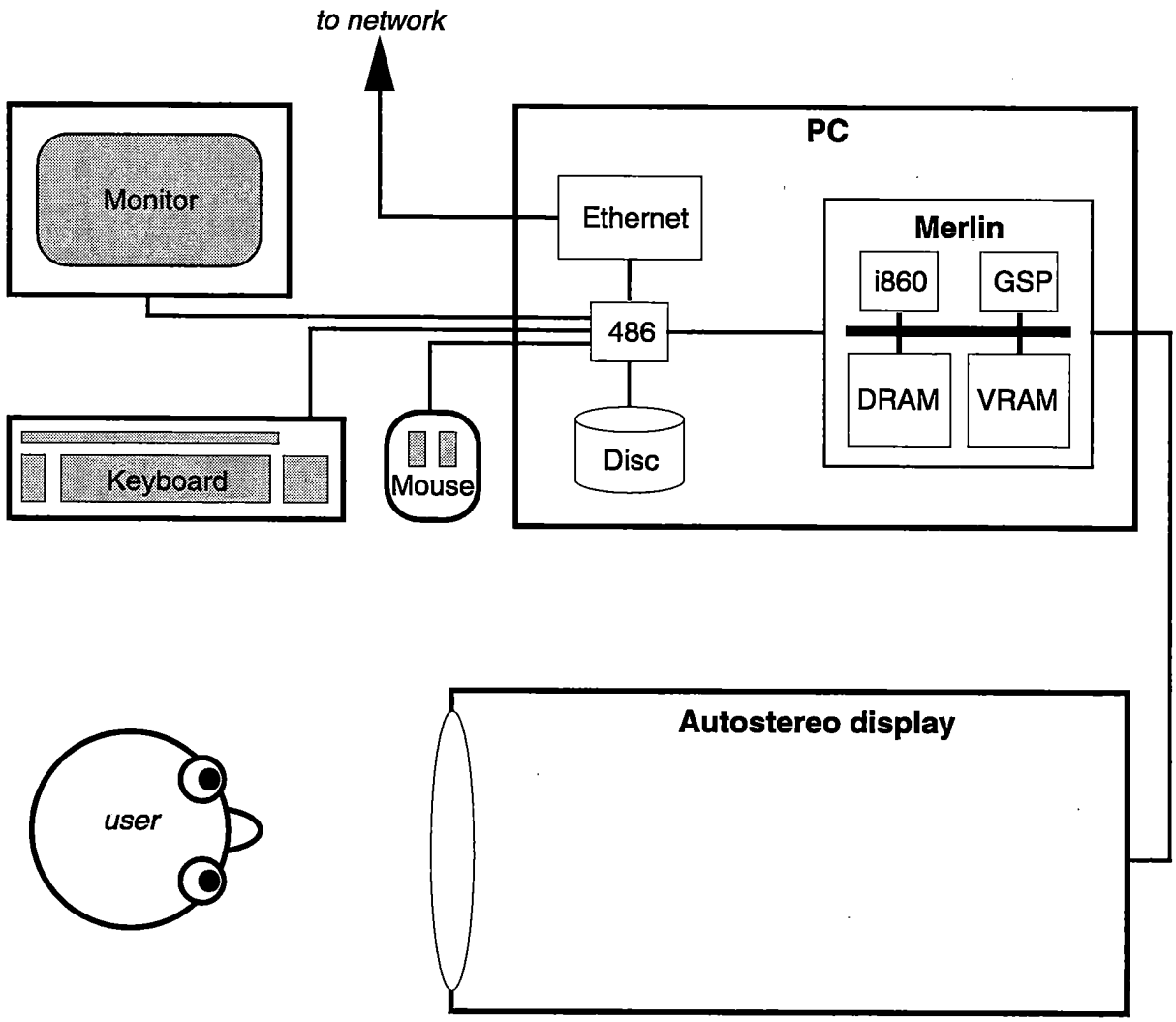


Figure 2.15: Hardware configuration of the experimental computer graphics support platform used with the autostereo display

2.5.2 The display device driver

The display device driver deals with the essential low-level housekeeping operations required to keep the autostereo display continuously supplied with video in-

put in the appropriate format. Its main task is to program the video hardware to output the sequence of frames which represent the views of an autostereo image in the order in which they are expected by the display. In addition to this it also handles other hardware-specific functions such as reading and writing colour palette intensities to the video output RAMDAC, and controlling screen blanking during idle update periods.

The display device driver software runs on the TMS34010 GSP, leaving the i860 CPU free for application tasks. It is composed of two main parts: an autostereo display list handler, and an application request processor. These will now be examined in more detail.

The autostereo display list handler runs as a background task on the GSP. It is an interrupt-driven routine which is triggered once per frame in the video output, at the refresh rate of the display. Its job is very simple: for each new frame of the autostereo image, determine which frame is to be output next and program the GSP hardware accordingly. This is achieved using look-up tables indexed by the current autostereo image format and frame sequence number. During the last frame of the autostereo image, the display list handler also outputs a ZSYNC pulse and then checks to see if a new set of views is ready to be displayed for the next autostereo image. The use of look-up tables makes it a simple task to add new autostereo image formats or to modify existing ones according to changing needs and makes the driver very flexible, allowing the same code to be used for all autostereo display formats currently supported.

The application request processor runs as the nominal foreground task on the GSP. It communicates with the application via a simple memory-mapped message passing protocol, processing requests from the application and (where appropriate) passing back results. It is responsible for passing updated images from the application to the display list handler, as well as providing the application with its own independent interface to hardware functions such as the palette colour look-up table (CLUT) and screen blanking. It is necessary to isolate the application in this fashion in order to prevent it from interfering with the display list handler's access to the video output hardware.

2.5.3 The applications interface

The lowest-level applications interface to the autostereo display is the MGL3D library. It builds on two independent subsystems: the device driver described in Section 2.5.2, and the 2D Merlin Graphics Library (MGL) supplied with the Merlin board (Datapath [1991a]). MGL3D allows the 2D MGL drawing functions to be applied to each view in an autostereo image. This library also handles double buffering of the autostereo image to allow one image to be displayed while the applica-

Table 2.3: Autostereo image formats and frame buffer requirements

Format code	Views	Resolution (h×v pixels)	Bits per pixel	Image size (Kbyte)	Double buffered on Merlin
L8	8	320×240	8	600	✓
L16	16	320×240	8	1200	✓
D16	16	640×240	8	2400	x
H6	6	640×480	8	1800	✓
H8	8	640×480	8	2400	x
L6C24	6	320×240	24	1350	✓
L6C8	6	320×240	8	450	✓
D6C24	6	640×240	24	2700	x
D6C8	6	640×240	8	900	✓

tion is preparing another. When the new image is ready, it is passed on to the device driver for display and the other buffer is used as the new drawing buffer. This prevents an observer from seeing an image in a partially complete (and possibly stereoscopically inconsistent) state while it is being rendered. Table 2.3 lists the memory requirements for all the autostereo image formats currently supported. Note that there is not sufficient VRAM in the Merlin's 4 Mbyte frame buffer to accommodate two separate image buffers for all of the autostereo image formats, which may influence the choice of autostereo format for some applications.

On top of MGL3D, further layers of software have been added to cope with 3D rendering in a manner that is independent of lower-level details concerning the autostereo image format being used, such as the number of views or the resolution of the image. The ASD (AutoStereo Drawing) library[6] has become the standard applications interface for 3D primitive rendering on the autostereo display used by members of the graphics research group here in the Computer Laboratory at the University of Cambridge. This library performs all the basic viewing geometry and autostereo projection calculations for point, line, and polygon primitives, but does not itself provide any routines for visible surface determination. Instead, low-level function hooks are provided which allow user programs to supply their own visible surface determination routines for the various primitives. This mechanism is used to test the visible surface determination algorithms described in Chapters 5 and 6.

[6]written by Neil Dodgson

2.5.4 The applications environment

The applications environment on the Merlin graphics board is of a fairly rudimentary nature. All applications are written in the C programming language and cross-compiled on the PC using the Portland Group C Compiler for the i860 (Portland Group [1991]). There is no operating system (as such) running on the Merlin board, and only one program may be running on the i860 at any one time. In addition, there is no built-in support for any standard input/output (I/O) operations for communicating with anything outside the Merlin board. Indeed, the only means of communicating with the board is via a 16-bit shared memory port which is addressable only from the PC.

To enable an application on the Merlin board to access peripheral devices attached to the PC, a special `host` program executes on the PC in parallel with the i860 application. A simple protocol for passing messages between the i860 application and the PC host allows requests for I/O operations (such as file handling, reading from the keyboard or mouse, writing to the console screen, and so forth) to be processed by the host on demand from the application. Any data which may need to be passed back to the application from the host as a result of such requests is returned in a similar manner.

In order to make the entire mechanism as transparent as possible to the i860 application programmer, these peripheral accesses have been encapsulated as functions which conform to standard C library specifications wherever possible. For example, a set of file handling operations have been implemented according to the specifications of the `stdio` library (Kernighan and Ritchie [1988]).

The primary advantage of such an approach is that it maintains a reasonable degree of compatibility with programs written for other platforms. A second advantage is that the application is isolated from the minutiae of the process of negotiating with the host to perform each I/O function. Its main disadvantage is that the interface between the PC and i860 is relatively slow (8 MHz), which can result in a significant degradation of application performance with certain repetitive, high frequency I/O operations.

2.6 Summary

In this chapter, the technological background of the work done in this thesis has been presented. A novel three-dimensional display device produced by colleagues at the University of Cambridge has been described, in terms of selected theoretical and practical aspects of its operation. An experimental computer graphics system to support the display has been developed, and an overview given of its implementation.

With the overall context established, the computer graphics background of this work may be introduced. An overview of the field of image synthesis is presented in Chapter 3, with particular reference to those issues which are most relevant to the type of three-dimensional display device described in this chapter.

3

Synthetic image generation

Synthetic image generation is a term which refers to the production of a digital image from a model held in a computer. This chapter is concerned with identifying and describing some of the issues involved in this process, both from a historical perspective and with a view to how existing knowledge in the field may be adapted or extended for use with an autostereo display device.

3.1 Background and terminology

For the purposes of this discussion, synthetic image generation is the process of producing a digital image from a three-dimensional model of an environment, described in terms of its geometrical properties and physical appearance. The digital image derived from this model represents a particular view of the environment as seen from a given viewpoint.

Generally such an image is intended for viewing on a flat, two-dimensional surface of finite size, such as a printed page or a CRT screen. Therefore the image may be obtained by finding the projection of the environment onto a planar viewing window positioned in front of the viewpoint, with the bounds of the viewing window corresponding to the bounds of the viewing surface. The task faced by synthetic image generation can thus be thought of as determining *what* is visible, *where* it projects to on the viewing window, and *how* it is to appear. Although these three aspects are often closely linked in practice, it may be useful to decompose the

problem in this manner. For a more comprehensive review of many of the issues discussed in the following sections, see Foley et al. [1990], chapters 14, 15 and 16.

3.1.1 What is visible?

Identifying what is visible is the *visible surface determination* problem[1]. A visible surface algorithm generally must consider what parts of the environment are potentially visible given a particular viewpoint, and then determine which of these parts are actually visible by considering their spatial relationships with each other. Assuming all the elements in the environment are opaque, this may be reduced to finding which part of the environment is nearest the viewpoint at each point inside the viewing window. This *minimum depth search* approach can be thought of as a description of a canonical visible surface algorithm, and provides the basis for many of the techniques developed to date.

Alternatively, geometric relationships between individual parts of the environment may allow the relative priority of one part with respect to another to be determined without resort to an explicit search for the nearest element. If this priority ordering can be applied in a hierarchical manner to all elements in the environment, considerable efficiency may be gained in the visible surface determination process. Indeed, it may be possible to produce a structured ordering for an environment which encapsulates these priority relationships in such a way as to be independent of the position of the viewpoint.

3.1.2 Where does it appear in the view?

It is often possible to considerably simplify the problem of what is visible by taking into account where each part of the environment projects to on the viewing window. Perhaps the simplest example of this is the clipping of the environment geometry to the *viewing frustum* formed by the pyramidal cone with its apex at the viewpoint and its sides passing through the edges of the rectangular viewing window, so that only those parts which project inside the viewing window are considered by the visible surface algorithm. It is possible to extend this notion to clip the environment geometry to within arbitrary depth bounds as well, further reducing the region to be considered. This is shown in Figure 3.1.

Considering the geometry of the perspective projection can further assist in the visible surface algorithm. For instance, the depth relationships between those parts of the environment which project to non-overlapping areas on the viewing plane may never need to be determined. Furthermore, identifying parts of the environ-

[1] also known as the *hidden surface elimination* problem

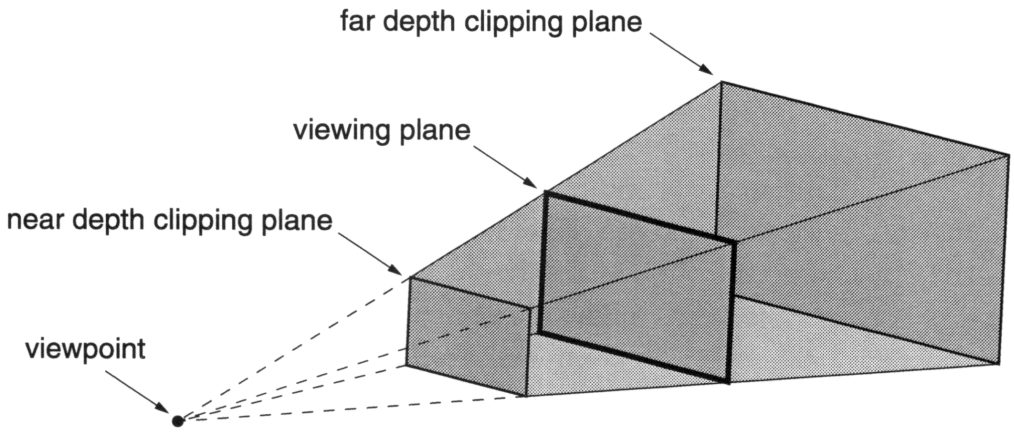


Figure 3.1: Clipping the environment geometry to the viewing frustum. Any part of the environment which falls outside the shaded region may be safely ignored by the visible surface process.

ment which do overlap but where one part can be shown to completely obscure another more distant part can also help simplify the problem. Structuring the model of the environment to reflect the spatial relationships between various parts of the environment can greatly assist in performing these kinds of tests.

3.1.3 How should it appear?

Having found out what is visible and where it should appear within the viewing window, all that remains is to determine how it should appear. For a digital image, this involves applying a suitable illumination model to compute colour and shading information for all relevant points in the image. These intensity samples should then be filtered in an appropriate manner to obtain an antialiased array of pixels which best represent the image on the intended display medium.

The computational effort which may be required to achieve this should not be underestimated. For many years practitioners in the field of image synthesis employed a variety of simplified illumination models and heuristic shading techniques to approximate the appearance of objects and materials according to empirical experience. In fact, the problem of how to realistically shade a surface, taking into account its relevant physical properties and sources of illumination (both direct and indirect) turns out to be full of subtleties and intricacies which far surpass the complexity of the visible surface problem. Similarly, antialiasing techniques have grown considerably more sophisticated in recent years as research has gained further insight into the nature of the problem.

3.1.4 Issues for multi-view stereo

As described in the preceeding sections, the task of synthetic image generation can be broken down into a number of related but separate components. It may be argued that of these components, the visible surface problem is the one which demands most attention for a multiple-view stereo display system. Assuming the shading techniques applied are capable of distinguishing the various different parts of the environment from one another, it is the relative positions of the visible surfaces in each view which allow the observer to perceive stereoscopic three-dimensional relationships which are the *raison d'être* of any true 3D display. An efficient means of visible surface determination is therefore of vital importance for a multi-view display and as such will provide the principal focus here. It must be remembered however that accurate perception of depth in a stereo image depends on an accurate representation of the environment as seen in each view, and thus some of the associated image quality issues concerning sampling and antialiasing techniques will also be reviewed.

3.2 Visible surface determination

Visible surface determination is the process of identifying which parts of the environment are visible through the viewing window from a given viewpoint. This entails finding not only those objects which fall within the viewing frustum, but also resolving their visibility relationships due to occlusion along a given line of sight. Thus while it is natural that any solution to the visible surface problem is highly view dependent, there is much similarity between the views in a stereo image which may be exploited. Here we briefly review some of the basic approaches to visible surface determination and a selection of conventional single-view visible surface algorithms before considering what alternative approaches may have to offer for a multi-view stereo image.

3.2.1 Approaches to visible surface determination

The visible surface determination problem has been addressed in a variety of ways by many researchers in the field of computer graphics. A review of ten of the best-known early algorithms may be found in Sutherland et al. [1974], which classified approaches as operating in *object space* (that is, to an arbitrary precision) or *image space* (that is, to a precision only as high as that required by the resolution of the displayed image). It is arguably more useful to consider this distinction as being between techniques which operate to *object precision* or *image precision* respectively

as suggested in Foley et al. [1990], pages 649–650 — a convention which is followed here.

3.2.1.1 Object precision approaches

Object precision algorithms produce a solution which is independent of the image representation in terms of resolution and which is thus scalable to any desired target resolution. The solution may be made to an arbitrary degree of accuracy — limited only perhaps by the precision of the numeric representation used — and thus typically involves a reasonably high level of algorithmic complexity. Object precision visible surface algorithms generally do not concern themselves with many of the practical aspects of converting the solution into an actual image for display, often ignoring issues such as illumination, shading, and anti-aliasing for pixellated (digital) images. Thus they remain essentially of theoretical rather than practical interest for the most part, although the field continues to attract research attention (Sharir and Overmars [1992]; Goodrich [1992]).

3.2.1.2 Image precision approaches

Image precision algorithms produce a solution which is intended for display at a specific pixel resolution. They often benefit from simplifying assumptions derived from this limited image resolution, and thus are generally of a lower algorithmic complexity than object-precision approaches. The techniques used for illumination, shading and antialiasing of the digital image are often closely bound to visible surface determination in image precision approaches for reasons of efficiency. Algorithms which fall into this category include Warnock's area subdivision algorithm, Watkins' scanline algorithm, and Catmull's Z-buffer algorithm (Sutherland et al. [1974]). Due to their pragmatic approach and conceptual simplicity they have become popular for use in a variety of situations, including real-time or interactive applications where hardware implementation is facilitated by their relative simplicity.

3.2.1.3 Hybrid approaches

Hybrid algorithms which combine elements of both object precision and image precision approaches have been developed which produce an image precision result based on object precision computations. Benefitting from both some of the simplifying assumptions possible with image precision techniques as well as the high accuracy of the object precision calculations (albeit at correspondingly high cost), they are generally of intermediate algorithmic complexity. Illumination, shading and antialiasing are generally closely bound to the algorithm. Algorithms described

to-date which fall in this category include priority-based techniques such as the painter's algorithm (Newell et al. [1972]) and Binary Space Partioning (BSP) trees (Fuchs et al. [1980]), as well as recursive ray-tracing (Whitted [1980]). However the potentially high cost of the object-precision part of these approaches tends to limit their application to areas where this cost is either acceptable or manageable to within certain limits.

3.2.2 Single-view visible surface algorithms

Despite much research into image synthesis over the past twenty years, it is perhaps surprising to note that the fundamental approaches to visible surface determination have changed so little since the classic review of the field by Sutherland et al. [1974]. In that time considerable experience in the implementation and application of these approaches has been gained, with the result that some approaches have gained wide acceptance and attracted further development, while others have not been so successful.

As computing power has increased and memory costs have fallen over the same period, so the size and complexity of the modelled environments which can be handled has increased. This has opened up new application areas such as computer-aided design, architectural modelling and entertainment which place an ever greater demand on the synthetic image generation techniques employed. This trend towards greater complexity has tended to favour those approaches which best cope with this increased load, both in terms of execution time and memory requirements. Arguably the two most popular methods for visible surface determination in use today are ray tracing (Whitted [1980]) and the Z-buffer (Catmull [1974]), albeit perhaps for different reasons. These are discussed in the sections which follow.

3.2.2.1 The Ray Tracing algorithm

Ray tracing (Whitted [1980]) is a conceptually simple algorithm based on the idea of following the path of light rays entering the eye of the observer back to their source in order to determine what is visible from that viewpoint. Visible surface determination is performed by selecting the nearest object met by each ray cast from the viewpoint through each pixel in the plane of the viewing window. A suitable illumination model is applied at the ray/object intersection point to determine the appearance of the object at that point and thus colour the pixel. A complete image may be built up pixel by pixel in this manner. This process is illustrated in Figure 3.2.

With this basic ray tracing technique in place, it is possible to significantly improve the quality of the resulting images by extending the use of ray casting be-

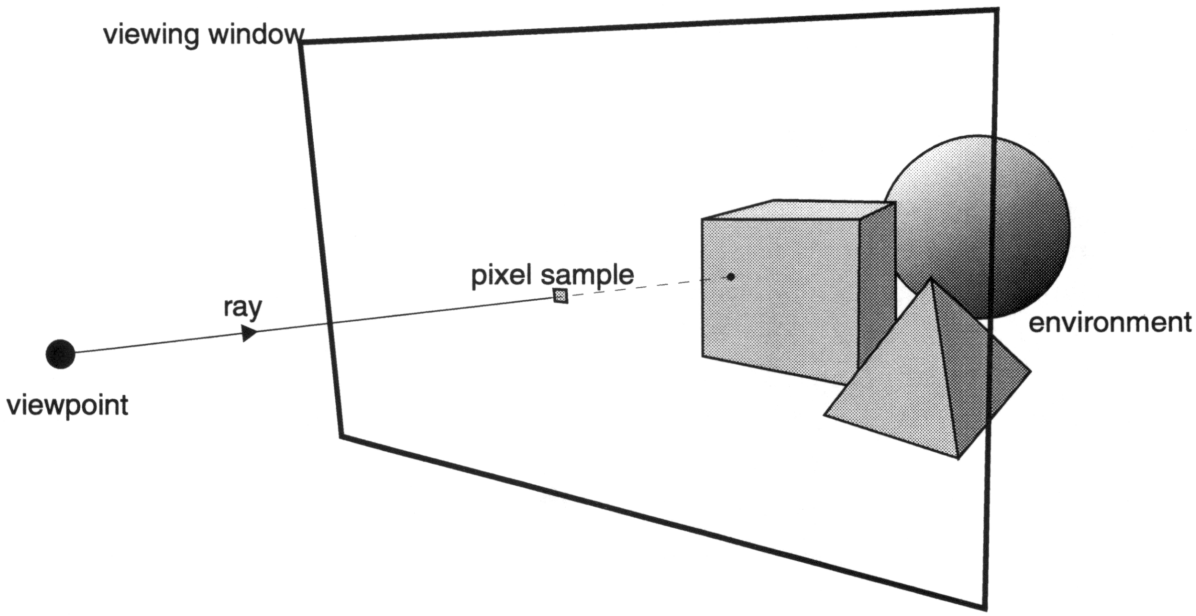


Figure 3.2: The Ray Tracing algorithm. The path of a light ray is traced back from the viewpoint through a point in the plane of the viewing window. The colour of the image at that point is determined by examining which object is hit first by the ray.

yond mere visible surface determination (Whitted [1980]; Cook et al. [1984]). By propagating further rays from each the ray/object intersection point, it is possible to readily simulate complex reflection, refraction and shadowing effects. By casting multiple rays through each pixel in slightly different directions and filtering the contributions from each ray, antialiasing may be performed. By distributing multiple initial rays over the surface of a conceptual lens, depth-of-field (optical focus) effects may be simulated. By sampling the object geometry with multiple rays per pixel as it changes position over time, motion blur effects may be simulated. All this is possible due to the power and generality of the fundamental ray/object intersection test.

Thus ray tracing provides the basis for a very flexible, high-quality image synthesis system, capable of simulating a wide variety of optical effects within the framework of a conceptually simple algorithm. However, the simple algorithm described is very demanding computationally, owing to the expense of the critical ray/object intersection test at the heart of the algorithm. It is possible to reduce this load by the use of hierarchical spatial object database structures (Rubin and Whitted [1980]; Glassner [1984]; Kay and Kajiya [1986]), yet image rendering times of scenes of even quite modest complexity are still typically measured in minutes rather than milliseconds. For this reason, ray tracing is generally favoured for use

where image quality is considered of prime importance and image generation rates compatible with interactive response times are not required.

3.2.2.2 The Z-buffer algorithm

The Z-buffer algorithm (Catmull [1974]) derives its name from its principal data structure. Just as the frame buffer for a raster display device contains a set of pixel values which represent the intensity to be displayed at each point in the image, so the Z-buffer contains a Z-value for each pixel which represents the depth of the object visible at that point. Each object is projected onto the plane of the viewing window and scan converted into individual pixels ready to be placed in the frame buffer. As this is done, the depth of the object at each pixel location is also computed. If the depth of the object at a particular pixel is nearer the observer than the depth represented by the Z-value stored in the corresponding position in the Z-buffer, then the object's pixel is considered to be visible in preference to the existing one. In this case, the object's pixel value is written to the frame buffer and the Z-value of the object's pixel is written to the Z-buffer. If the depth of the object is more distant than the currently-stored Z-value at that pixel, then the existing pixel is considered to occlude the object at that point and neither the frame nor Z-buffers are modified. The Z-buffer algorithm can thus be thought of as an image-precision version of the canonical minimum depth search visible surface algorithm described in Section 3.1.1. Figure 3.3 illustrates the way in which objects are projected onto the plane of the viewing window for processing by the Z-buffer algorithm.

Although the Z-buffer algorithm is capable of dealing with any three dimensional object in theory, perhaps the simplest to handle is the planar polygon. Scan conversion of a planar polygon is straightforward and well-suited to linear interpolation using only integer operations. This and its fixed-size memory overhead (for the actual Z-buffer itself) are features which lend themselves to implementation in hardware or software. More complex objects may be handled by first decomposing them into a suitable polygonal approximation, or it may even be possible to process them directly if Z-values can be readily computed.

With the basic visible surface determination algorithm in place, the question of image quality may be addressed. Antialiasing may be achieved in a number of ways, which generally may be classed as being either *subpixel area based* (Carpenter [1984]) or *multiple point sampling* (Fuchs et al. [1985]) in nature. Refraction according to the laws of optics is difficult to simulate using a Z-buffer, although simple (non-refractive) transparency is possible by blending the colours of overlapping translucent layers with the background (Mammen [1989]) or through the use of so-called "screen-door" approximations (Fuchs et al. [1985]). Reflection is similarly tricky to deal with accurately, but careful application of environment map-

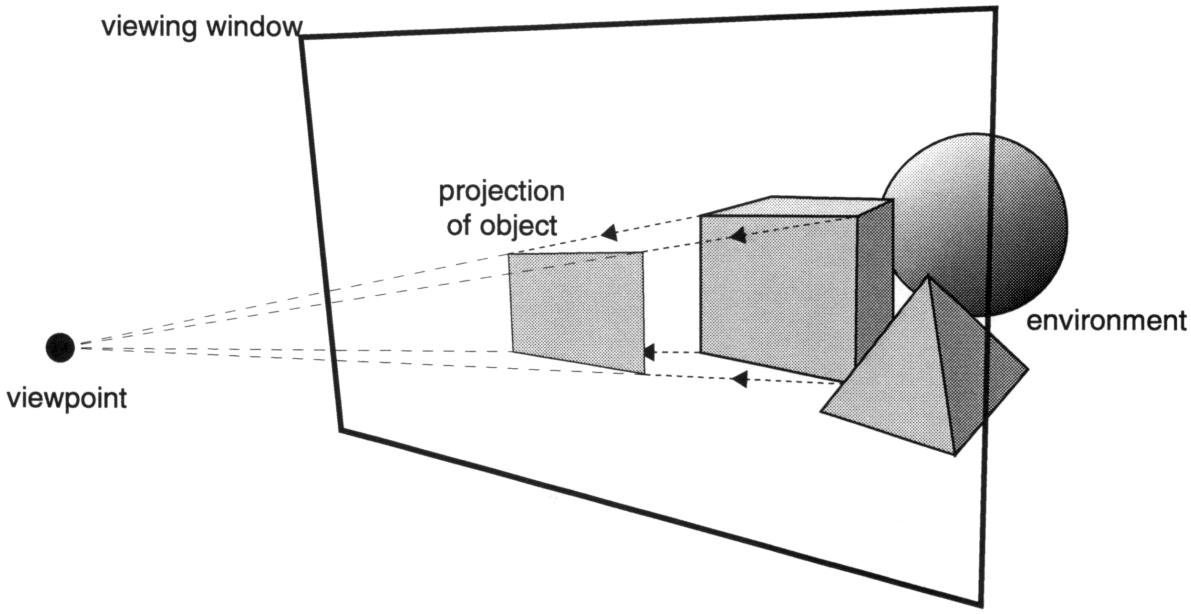


Figure 3.3: The Z-buffer algorithm. An object is projected onto the plane of the viewing window, where the depth at each point in the projected object is used to determine its visibility compared with objects already processed.

ping techniques (Blinn and Newell [1976]) can produce useful results in certain circumstances. Shadows from point light sources may be simulated by performing a visible surface operation from the viewpoint of the light source and then reprojecting the results back into the original view (Williams [1978]). Depth-of-field effects may be approximated in a separate post-processing operation (Potmesil and Chakravarty [1981]), utilising the depth information in the Z-buffer to control the degree of blurring to each pixel. Motion blur may be simulated by integrating multiple images sampled at different points in time[2] (Haeberli and Akeley [1990]). Furthermore, it is possible to modify the basic Z-buffer algorithm to make use of hierarchical spatial data structures to significantly accelerate the rendering and antialiasing of complex environments (Greene et al. [1993]; Greene and Kass [1994]).

The Z-buffer approach thus provides a simple method for visible surface determination that is suitable for efficient implementation in hardware and yet sufficiently flexible to be applied to a variety of tasks. While it is undoubtedly not as accurate or as readily adaptable as ray tracing for the simulation of many optical effects, it is nonetheless possible to adequately approximate most of these effects given sufficient ingenuity and effort. For these reasons, the Z-buffer is generally

[2] a similar technique may be used to obtain more accurate depth-of-field effects by integrating multiple images sampled from a number of different but closely-spaced viewpoints

favoured where rendering speed and efficiency are valued above strict notions of image realism.

3.2.3 Multi-view visible surface algorithms

Until recently, visible surface algorithms which address the problem for multiple views have been relatively rare. Indeed, as monoscopic viewing has been the dominant mode for synthetic images, there may have seemed to be little need to consider such a problem. Typically, the approach taken to generate multiple views of an environment from different viewpoints has been simply to repeat the procedure used to obtain a single view independently for each view, regardless of any duplication of effort due to similarities between the views (Hodges [1992]).

Perhaps the closest to a multi-view approach to synthetic image generation has come from the field of computer generated animation. An animation sequence requires dozens of images per second of viewing to produce a smooth, continuous appearance of movement. Each successive image necessarily appears very similar to its predecessor so that a human observer can readily track the changing appearance of the objects in the scene from one frame to the next, thus conveying an impression of motion rather than a sequence of disjoint images.

These similarities between successive frames in an animated sequence of images and the potential efficiency gains for visible surface determination to be made by taking advantage of them was noted by Sutherland et al. [1974], with particular reference to Schumaker's *a priori* list priority algorithm. A hidden surface algorithm utilising frame-to-frame coherence was proposed by Hubschman and Zucker [1981], which computed incremental modifications to the visibility status of convex objects due to movement of the viewpoint from one frame to the next. Preprocessing of geometric relationships between objects in a static environment with BSP trees was used to accelerate the list priority visible surface process for a changing viewpoint by Fuchs et al. [1980]. Glassner [1988] describes a hybrid technique which attempts to combine an adaptive space subdivision with bounding volumes in the context of a 4-dimensional spacetime model to speed up ray tracing of large animation sequences. Chapman et al. [1991] describe an adaptation of the ray tracing algorithm which attempts to compute ray-object intersections over a specified time interval for moving objects.

While a sequence of views in an animation sequence can be thought of as snapshots of an environment changing in both space and time, a stereo image represents different views of the same environment at the same point in time. Thus an even greater potential exists to capitalise on the coherence between views in a stereo image than in an animated sequence of images, as only the viewpoint changes between one stereo view and the next while objects remain in the same positions in

the environment.

In recent years a variety of algorithms specifically intended for generating stereo images have been reported. Papathomas et al. [1987] present an algorithm for efficiently generating stereo views of meteorological voxel data in a point cloud display format. Adelson et al. [1991] describe stereo adaptations of conventional techniques for polygon rendering, Gouraud shading, line and polygon clipping, and back-face removal, as well as considering the use of Z-buffer, BSP tree and scanline hidden surface elimination algorithms for two-view stereo. Both these approaches essentially involve conventional techniques adapted and optimised to some degree for application to stereo image generation.

Instead of merely optimising conventional rendering of separate stereo views, an alternative approach that has been explored by a number of researchers is to attempt to generate an approximation of one view by reprojection of another with respect to the new viewpoint. Guo et al. [1988] describe a method for rapidly generating a large number of holographic views based on reprojection of points in a so-called "Position Oriented" (PO) multiframe buffer as seen from each separate point of view, where the PO buffer is created in an initial ray casting preprocessing step. Ezell and Hodges [1990] present an algorithm for inferring one ray-traced stereo view from another by a similar (although independently-developed) reprojection technique, with additional heuristics applied to attempt to identify and retrace any errant pixels in the inferred view. This technique was adapted to ray tracing of implicitly defined functions by Devarajan and McAllister [1991]. Adelson and Hodges [1993] refined the stereo reprojection algorithm described by Ezell and Hodges [1990] by developing a more rigourous solution to the errant pixel detection problem and considering some of the effects of multiple rays per pixel for antialiasing as well as reflection and refraction. Harrison and McAllister [1993] suggest that simple reprojection of a single Z-buffered image for other viewpoints without attempting to correct errant pixels might be acceptable for interactive applications, where rendering speed and thus response time is considered more important than high image quality.

All these reprojection methods are based on the idea that a point sampled image with depth values associated with each pixel represents a sparse approximate model of the environment from which it was produced (see Figure 3.4). The underlying assumption is that the task of identifying and correcting any errant pixels in the reprojected view is less computationally expensive than simply re-rendering the environment from the new viewpoint using conventional techniques.

Although not strictly intended for stereo images, similar ideas have been explored by other researchers. Chen and Williams [1993] propose a method for generating multiple views of an environment as seen from a large number of closely-spaced viewpoints by reprojecting and interpolating between a smaller number

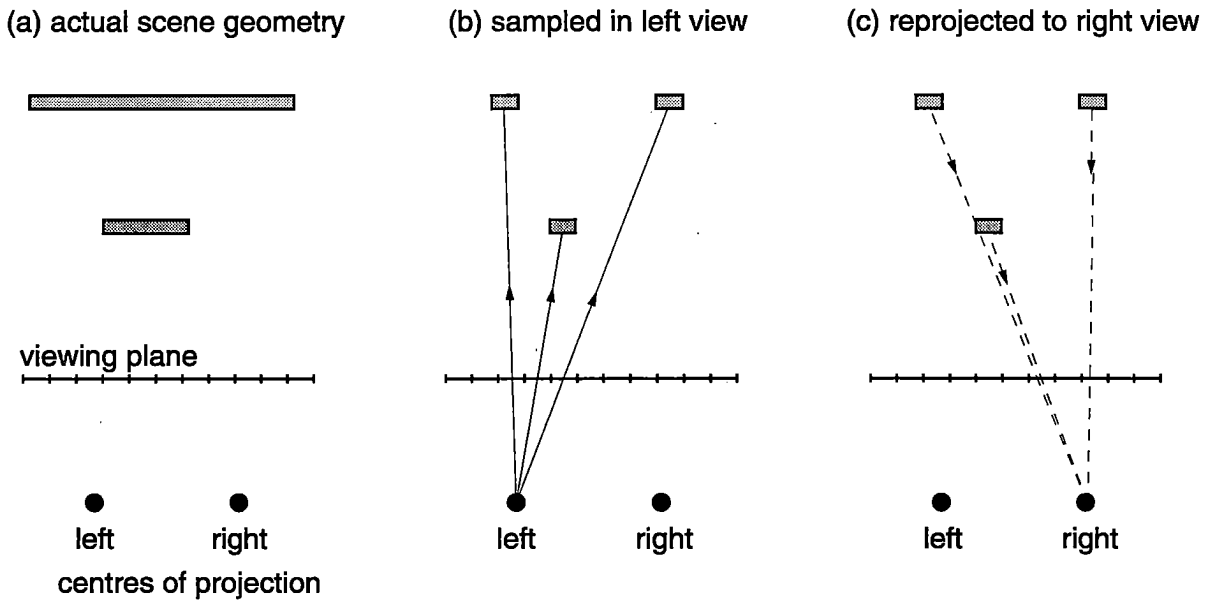


Figure 3.4: Reprojecting pixels from one view as seen from another

of known views of the environment using image morphing techniques. A similar reprojection and interpolation technique used to accelerate the production of ray traced animation sequences is described by Ward [1994], except that errant pixels may be optionally re-traced rather than interpolated, in a manner similar to the method used by Ezell and Hodges [1990].

3.3 Antialiasing

Antialiasing of synthetic images is a generic term which covers a variety of techniques used to attenuate objectionable image artifacts due to the representation of the image as a discrete array of pixels. Such artifacts may include smooth edges appearing jagged, small details missing or incorrectly represented, or smoothly moving objects appearing to jump unnaturally from one position to another in an animated sequence of images (Crow [1977]). In this section a selection of conventional antialiasing techniques will be reviewed from a practical viewpoint before considering some of the issues concerning antialiasing of stereo images in particular.

3.3.1 Antialiasing in theory

Although it is not the intention of this section to describe in detail the theory behind antialiasing of synthetic images, it is useful to define some basic terms which may be used to characterise the process in general. For a more detailed review of antialiasing and sampling theory applied to digital images, see Foley et al. [1990], section 14.10, on which much of this section is based.

A synthetic image is made up of a finite array of discrete pixels. Each pixel can be assigned one of a given range of discrete values, corresponding to different intensities and/or colours as seen in the displayed image. In contrast to the *discrete* nature of the synthetic image, the model of the environment from which the image is derived may be considered to be a *continuous* representation. The task of image synthesis may thus be seen as the process of converting this continuous representation into an appropriate discrete approximation which best represents a particular view of the environment.

The process of computing a discrete approximation of an image is known as *sampling*, and each discrete component of the approximation is referred to as a *sample*. The samples which make up an image are used by the display hardware to *reconstruct* a continuous approximation of the image for viewing by an observer. This process of sampling followed by reconstruction almost invariably involves a loss of information for most synthetic images. This may in turn lead to errors of interpretation by an observer due to misrepresentation of elements in the image.

The loss of information may be attributed to inadequate sampling or deficiencies in the reconstruction. Information lost in the sampling step cannot be recovered in subsequent processing. The best that can be hoped for is that the loss may be made as unobtrusive as possible. There is little that can be done in the image synthesis process to improve the quality of the final reconstruction of the image, as this is performed by the display device hardware. We must therefore focus our attention on improving the sampling instead.

If insufficient samples are taken to allow the accurate reconstruction of the image, aliasing may result and elements in the image may be misrepresented or omitted. Unfortunately, most synthetic images contain elements which theoretically require an infinite sampling rate for their correct reconstruction. It is possible however to reduce the sampling rate required by judicious application of *filtering*, which allows subsequent reconstruction of a suitable approximation of the intended image. This is the general approach to antialiasing for image synthesis.

3.3.2 Antialiasing in practice

A selection of some of the most widely-used antialiasing techniques in practice are summarised and compared here. Almost all of these methods apply filtering only after sampling and thus are less than ideal as they fail to guarantee that the image contains no frequency components too high for the sampling to faithfully represent. Instead the general approach taken is to attenuate the objectionable artifacts of aliasing as much as possible using combinations of various sampling and filtering techniques.

3.3.2.1 Point sampling

The simplest approach to sampling is *point sampling*. This method effectively interrogates the computer's model of the environment as seen through an infinitely small point in the image to obtain a sample value. The simplest algorithms for image synthesis take one point sample at the centre of each pixel and directly display the resulting image without any filtering. Aliasing in such an image may be severe, with all the classic artifacts such as jagged edges and missing detail (see Figure 3.5).

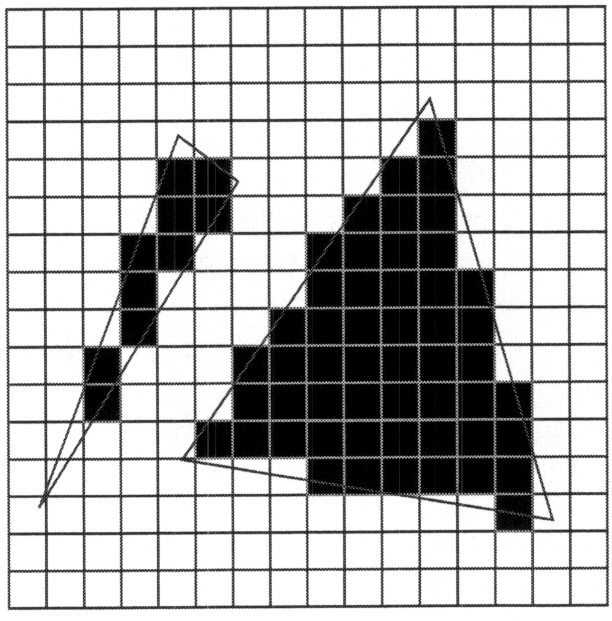
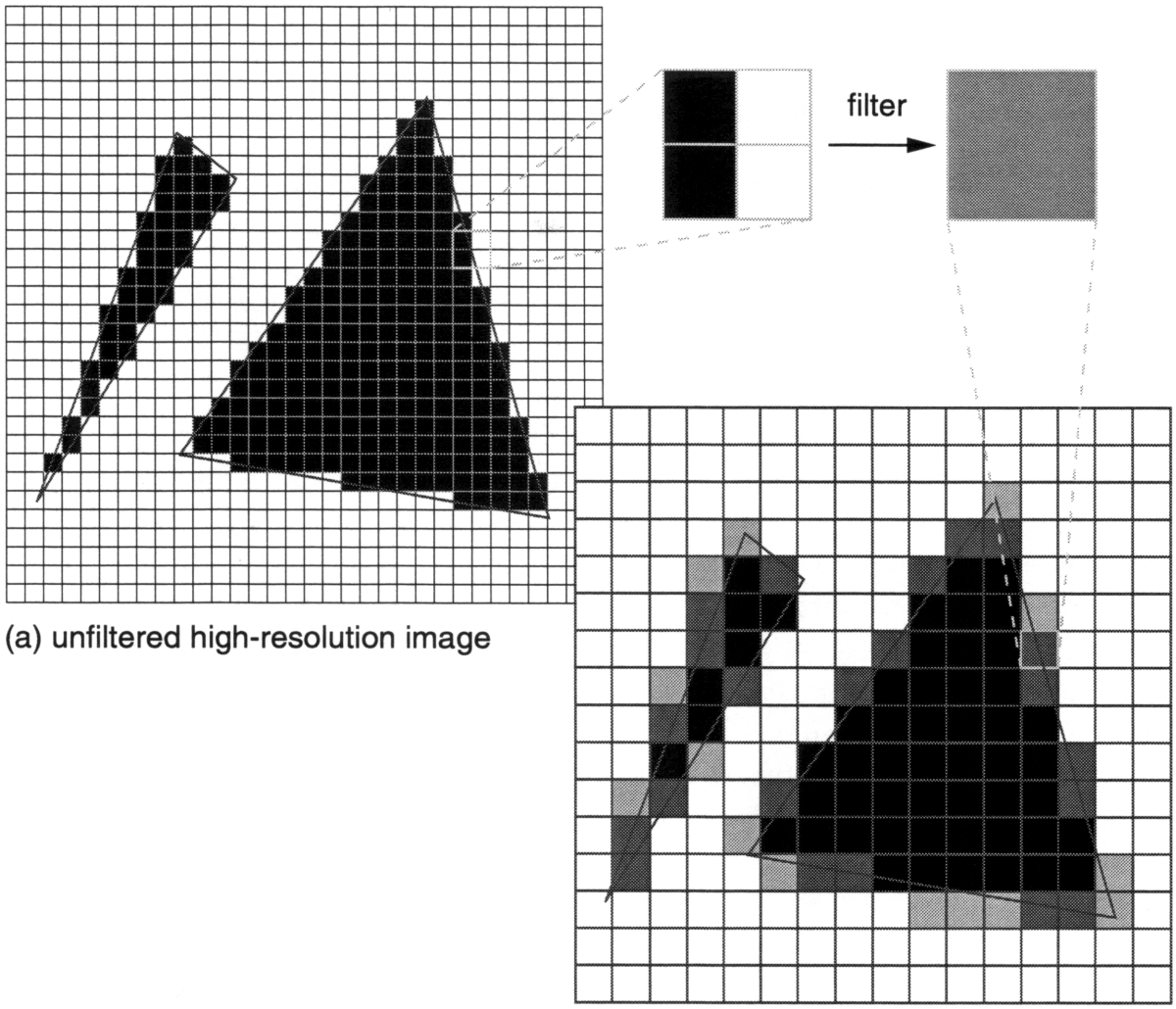


Figure 3.5: Common aliasing artifacts. Note the jagged appearance of the edges and the small details not represented by the sample points.

3.3.2.2 Supersampling

A much better image may be produced by taking more than one sample per pixel and filtering the resulting samples to form the image. This technique is known as *supersampling*. At its simplest, it may be considered to be similar to increasing the resolution (and thus the sampling rate) and filtering the result. See Figure 3.6.



(a) unfiltered high-resolution image

(b) filtered image

Figure 3.6: Supersampling. Increasing the density of sample points allows the shape of the object to be reproduced with greater accuracy and captures smaller details in the unfiltered high-resolution image (a) than in Figure 3.5, which are subsequently represented in the filtered image (b).

Supersampling is a simple example of a *postfiltering* method, that is the filtering is performed after sampling. Although generally quite effective and straightfor-

ward to implement, its major drawback is that there may be no way of determining whether a given set of samples adequately represents all elements in any given image. Important features on some elements may not fall under any sample location, and the accuracy of positional information regarding any given element will be limited to the distance between sample points. Furthermore certain sampling strategies tend to be more susceptible to certain kinds of image artifacts than others. For example, if the sample locations within each pixel are arranged on a regular rectangular grid, the correlations between sample points may still result in visible aliasing or drop-outs of nearly horizontal or nearly vertical features in the image. This issue is discussed further in Section 3.3.2.4.

3.3.2.3 Area sampling

Area sampling attempts to address some of the problems of point sampling by integrating each sample over a well-defined area rather than merely at a single point (see Figure 3.7).

If this integration is performed analytically at object precision before the sample is obtained, the method is a *prefiltering* technique. Prefiltering is attractive from a theoretical point of view as it provides an object-precision approximation to the image which is capable of being faithfully sampled and reconstructed. However it is generally considered to be practical only for images with low visual complexity (Crow [1981]) due to the expense of object precision computations.

Approximate area sample estimates may be obtained more efficiently using non-analytic postfiltering techniques. Discrete approximations to unweighted area sampling using bit-masks to represent pixel coverage at a specific subpixel resolution are much less costly than analytic alternatives and are capable of producing good results (Carpenter [1984]; Fiume et al. [1983]), despite the simple single-pixel box filtering employed. A more sophisticated technique which provides a discrete approximation to weighted area sampling with an arbitrary filter greater than one pixel in size has also been reported (Abram and Westover [1985]).

However it is possible to introduce errors in both visibility and area coverage estimates where multiple primitives overlap within a single pixel, due to the finite resolution of the regular subpixel sampling grid and the methods used to estimate visibility within the pixel. Techniques which specifically address these deficiencies are described by Schilling [1991] and Schilling and Straßer [1993] which further reduce visible aliasing and drop-out artifacts, albeit at considerable cost in increased complexity. These approximate area sampling methods share some of the elements of both the analytic prefiltering and the supersampling postfiltering approaches, trading off accuracy against complexity.

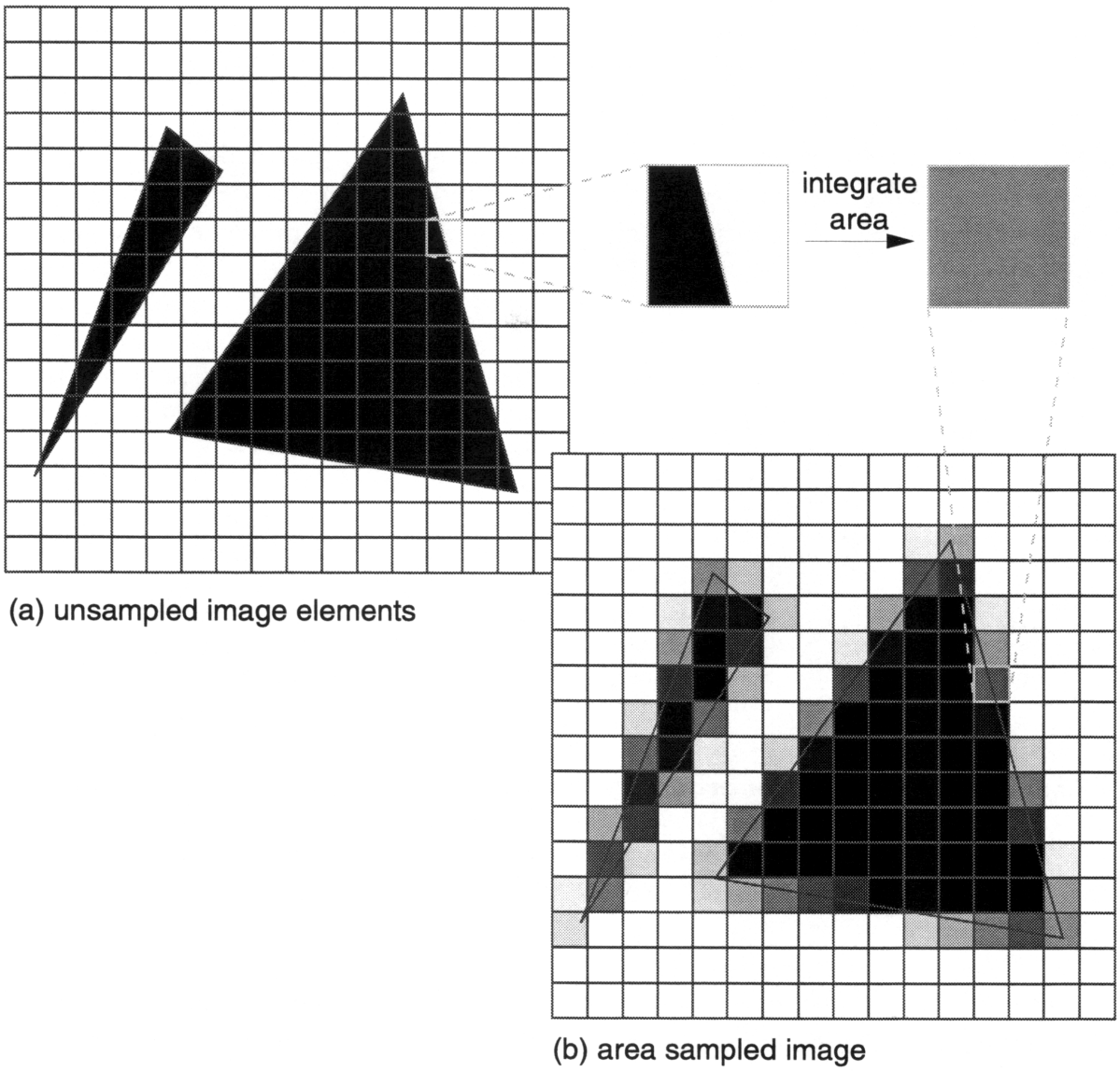


Figure 3.7: Area sampling. The area occupied by each element in (a) is precisely represented by analytic integration and thus each element's contribution to the pixel is faithfully reproduced in (b), in contrast to supersampling where very small details can still be missed by the sample points (see Figure 3.6).

3.3.2.4 Non-uniform sampling

Non-uniform sampling is an alternative to supersampling on a regular array of sample points which uses an irregular sampling pattern instead. Sampling at an inadequate rate on a regular grid may produce objectionable aliasing artifacts in the form of coherent false patterns. On the other hand, sampling at an inadequate rate in an irregular pattern and filtering the results tends to produce featureless noise which the human visual system is more tolerant of. Figure 3.8 compares uniform and non-uniform sampling patterns. Non-uniform sampling effectively performs a type of area sampling by Monte Carlo integration, and thus produces images of similar visual appearance to area sampling but generally at a much lower computational cost.

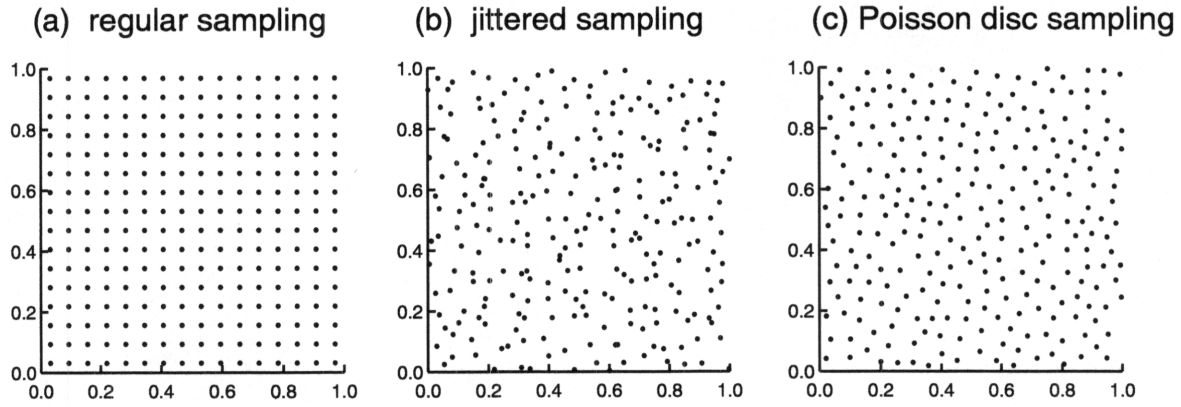


Figure 3.8: The advantages of non-uniform sampling. (a) shows a uniform regular sampling pattern, while (b) and (c) show two different forms of non-uniform sampling. Note how (c) manages to achieve more even coverage of the sample space than (b), while still avoiding the regular pattern of (a).

Also known as *stochastic sampling* (Cook [1986]; Dippé and Wold [1985]), the technique may potentially be used with any point sampling method. The principle behind non-uniform sampling has shown to be readily applicable to a variety of other image synthesis problems apart from spatial antialiasing, including motion blur, depth of field, penumbras (soft shadows), translucency, and fuzzy reflections (Cook et al. [1984]). Similar stochastic methods have also been applied to image synthesis problems in much wider contexts (Kajiya [1986]). Various sampling and filtering strategies have been described using non-uniform sampling, which attempt to optimise particular aspects of the procedure (Lee et al. [1985]; Mitchell [1987]; Mitchell [1991]; Kirk and Arvo [1991]). Non-uniform sampling has also been used as the basis for a number of multi-pass progressive refinement rendering tech-

niques (Fuchs et al. [1985]; Mammen [1989]; Painter and Sloan [1989]; Haeberli and Akeley [1990]; Barkans [1991]).

3.3.3 Antialiasing stereo images

Point sampled views in a stereo image are prone to all the same aliasing artifacts of point sampled non-stereo images described in Section 3.3.2.1. Stereo images may suffer from other aliasing artifacts apart from these however, including *depth aliasing* and *noncorrespondence aliasing* (Reinhart [1992]).

Stereo depth perception is produced by the perceived horizontal disparity between conjugate points in the views seen by the left and right eyes of an observer. If the views are point sampled, the actual position of a given point in the each view is quantised to the nearest integer pixel locations, resulting in a positioning error of up to ± 0.5 pixels in each view. When the views are subsequently viewed stereoscopically, the potential error in the perceived disparity is the sum of the individual positioning errors in each view, that is up to ± 1 pixel. This error in the stereo disparity leads to a corresponding error in the perceived depth of the point. Depth aliasing occurs when the depth of a point cannot be represented precisely by an integer pixel disparity and it is quantised to the nearest available disparity value. For example, Figure 3.9 shows the maximum potential relative depth uncertainty due to a single-pixel error in stereo disparity for an image on the autostereo display at various horizontal resolutions.

The aliasing artifacts caused by point sampling, such as the appearance of jagged edges or missing detail, are highly dependent on precisely where the point samples are taken in relation to the sampled element. Noncorrespondence aliasing may occur when different artifacts are present in the views seen by the left and right eyes. These differences may degrade the quality of the stereo image and interfere with the perception of stereoscopic depth in the image, possibly reducing the ease or speed with which the image may be fused stereoscopically.

Both of these problems may be alleviated by simply applying conventional antialiasing techniques to the view which make up the stereo image. In the same manner in which antialiasing can help increase the apparent resolution of a non-stereo image, so it can be used to increase the apparent depth resolution of a stereo image and thus reduce apparent depth aliasing artifacts. Similarly, as aliasing artifacts are attenuated, so noncorrespondence between stereo views is also diminished.

While conventional antialiasing techniques can be applied to views in a stereo image to reduce artifacts present both within each view and in the stereoscopic relationship between the views, it is not clear whether it may be possible to make any use of stereo coherence to make the process more efficient. This is because antialiasing is inherently view-dependent: the same object viewed from two different posi-

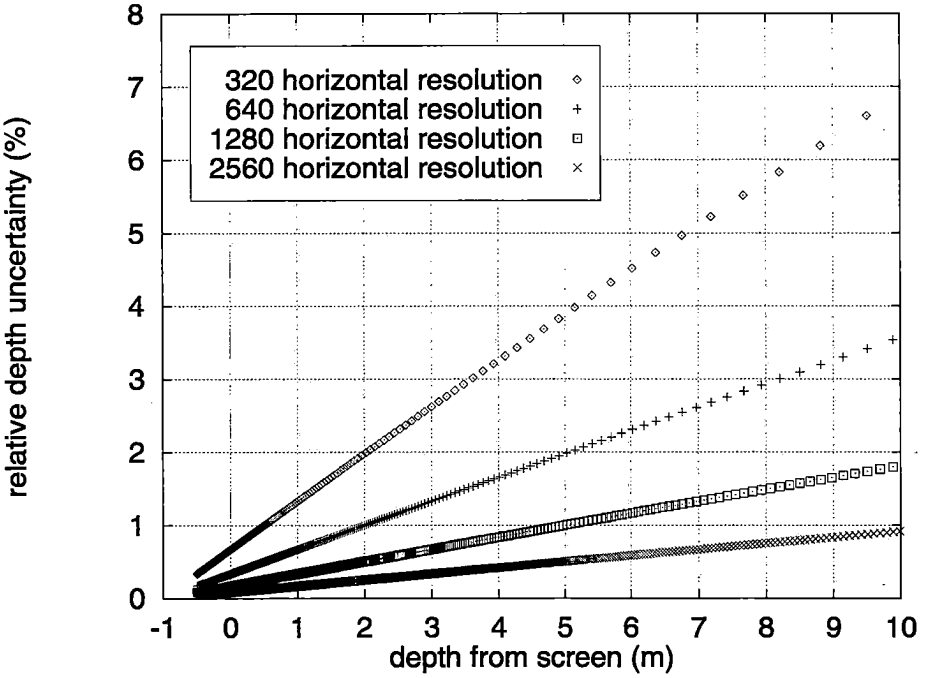


Figure 3.9: Relative stereo depth uncertainty due to single-pixel disparity errors for an image on the autostereo display

tions will almost always exhibit different aliasing artifacts in each view, depending on the relative position of the projection of the object with respect to the sample points used in each view. Thus the computations necessary to attenuate these artifacts are also likely to differ in each view, and stereo coherence may not be applicable.

3.3.4 Antialiasing multi-view autostereo images

A new problem with the discrete nature of the views in a multi-view autostereo image is that it becomes possible for the observer to perceive the boundaries between viewing zones as the head is moved from side to side. Parts of the image may appear to "jump" across the screen, combined with a visible "wiping" effect as the eye makes the transition from one zone to the next. This artifact becomes more pronounced the greater the stereo disparities between the views. These increased disparities may be either the result of larger spacing between the viewpoints, or due to image elements being portrayed at large depths from the view plane. A characterisation of this problem in terms of sampling theory as applied to holographic stereograms is presented by Halle [1994].

One possible solution to this is to limit the spacing between viewpoints, by either limiting the overall stereo field-of-view of the image, or by increasing the number of views used to cover the same field-of-view. Unfortunately, it is often not possible to control either of these parameters in practice, owing to constraints of the display hardware used. Another possibility is to treat this artifact as a form of stereo "view aliasing" and apply an appropriate filter to the image, so as to make each view better represent what is seen from the corresponding region of space rather than just at a single point. This latter approach, similar to that described by Halle [1994], is discussed further in Section 5.12.2.

3.4 Summary

In this chapter, some of the issues surrounding synthetic image generation have been introduced, with particular reference to their application to images produced for viewing on a multi-view stereo display. It has been suggested that the visible surface determination component of the synthetic image generation problem is the most important for multi-view stereo systems, and some of the prior work in this area has been reviewed. Now it remains to investigate further the possibilities for taking advantage of the similarity between views in a stereo image in the visible surface determination process to reduce the computational effort required to generate multi-view stereo images. The key to understanding how this can be achieved may be found by examining the viewing model which underpins so much of the synthetic image generation process. This is the subject of Chapter 4.

4

Viewing model

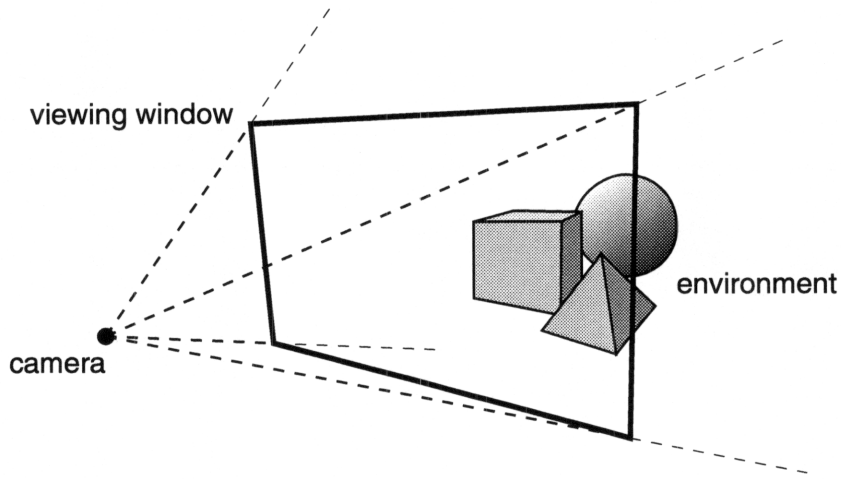
The *viewing model* for a display is a model of the relationship between the human observer and the image visible on the display. The viewing model is of fundamental importance to the entire synthetic image generation process, which uses its geometric and spatial characteristics as the basis for determining how objects are represented in the final image. Before describing the proposed multi-view autostereo viewing model, existing monoscopic and two-view stereoscopic 3D viewing models are reviewed to highlight the similarities and differences between the new display and conventional devices for rendering purposes.

4.1 Conventional monoscopic 3D viewing models

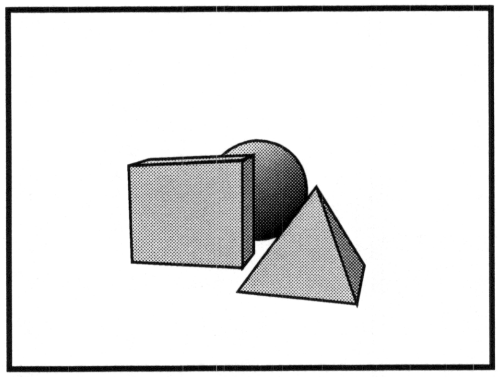
4.1.1 A camera viewing model

One conventional 3D viewing model in common use with planar CRT displays is the *camera model* (Foley et al. [1990], section 6.3). In this model, a virtual camera takes the place of the observer's eye and a virtual viewing window takes the place of the display screen. The camera points towards the viewing window and the image seen by the observer on the CRT screen is based on what the camera can see in this window. This is illustrated in Figure 4.1.

In general, the exact location of the observer's eye in relation to the display screen is not known. In the absence of such information, the assumption commonly



(a) the relationship between the camera, the viewing window and the environment



(b) what the observer sees on the display screen

Figure 4.1: Monoscopic camera viewing model

made is that the observer is situated in front of the centre of the screen and thus the virtual camera is positioned in front of the centre of the display window. The distance of the observer from the screen (which is also not generally known) together with the physical size of the display screen determines the geometrical field-of-view of the image seen by the observer. The implications of this latter issue are discussed further in Section 4.1.4.

4.1.2 Reproducing the monoscopic camera view

The task of reproducing on the CRT screen what is visible in the frame of the window from the camera's viewpoint is the responsibility of the rendering procedure.

For planar displays, this may be readily achieved by projecting visible elements of the scene onto the viewing window.

Two common planar projections are the *orthographic* and the *perspective* projections. In orthographic projection[1], object points are projected perpendicularly onto the viewing plane, thus preserving relative distances between object points regardless of distance from the viewing plane. This *size-preserving* feature is useful in many applications where scale is important, such as computer-aided design (CAD). In contrast, the perspective method projects objects onto the viewing plane through a single point — the *centre of projection* — which has the effect of making more distant objects appear smaller than ones nearer to the centre of projection. Of the two, perspective projection most closely approximates how the human eye views the world, and thus is most often used to portray images of real or virtual worlds in computer-generated images. Unless specified otherwise, only perspective projection will be considered in the following discussion.

4.1.3 Perspective projection geometry

The geometry of a simple perspective projection is shown in Figure 4.2. A left-handed viewing coordinate system is assumed. The camera position is used as the centre of projection which, in the absence of any other information about the location of the observer in front of the display, is taken to be perpendicular to the centre of the viewing window at a distance d from the window on the negative z-axis. Thus the centre of projection is $C(0, 0, -d)$ and a point $P(X, Y, Z)$ in the scene is projected onto point $Q(x, y, 0)$ in the viewing plane. The coordinates of Q are given by:

$$x = \frac{X \cdot d}{d + Z} \tag{4.1}$$

$$y = \frac{Y \cdot d}{d + Z} \tag{4.2}$$

Alternative views of the scene may be obtained by moving the camera viewpoint and viewing window relative to the scene. To simplify the projection geometry, an equivalent operation is to move the scene relative to the viewpoint and viewing window. This establishes the location of each element in the scene in relation to the camera's frame of reference and is achieved by applying the *viewing transformation* to each point in the scene.

[1] also known as *parallel projection*

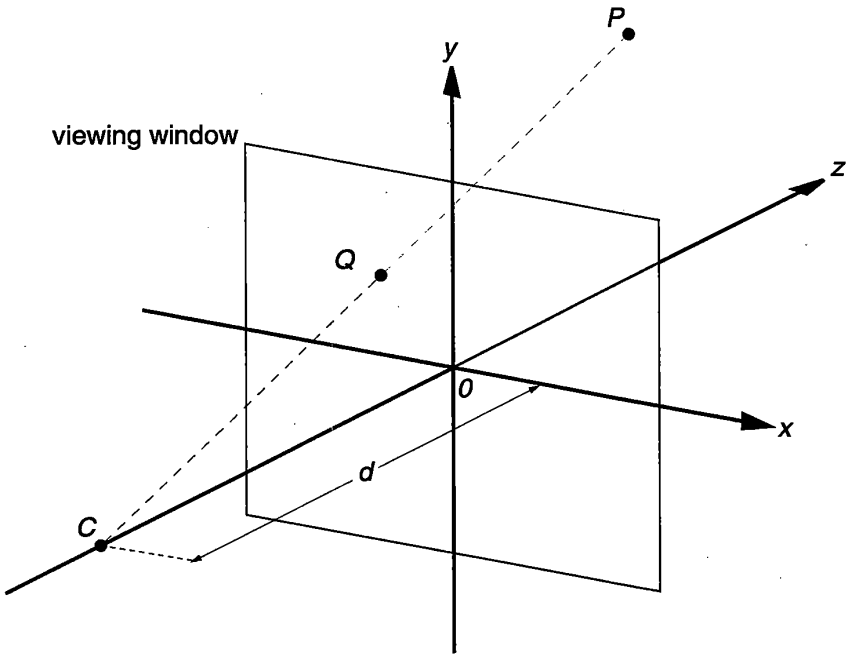


Figure 4.2: Monoscopic perspective projection geometry

4.1.4 Monoscopic field-of-view

More of the scene may be made visible by increasing the size of the viewing window, which effectively increases the camera's field-of-view. In the real world, a display screen of fixed physical size means that the viewing window image will have to be scaled down to fit. To maintain correct perspective viewing, the observer should adjust his position in relation to the display screen so as to match the perceived field-of-view of the image with the field-of-view used to produce it, although this is rarely done in practice.

The relationship between viewing window size and field-of-view is shown in Figure 4.3. Assuming the same viewing arrangement as described in Section 4.1.3, the horizontal (w_h) and vertical (w_v) viewing window sizes are expressed in terms of the horizontal (f_h) and vertical (f_v) camera field-of-view as follows:

$$w_h = 2d\tan(\frac{f_h}{2}) \tag{4.3}$$

$$w_v = 2d\tan(\frac{f_v}{2}) \tag{4.4}$$

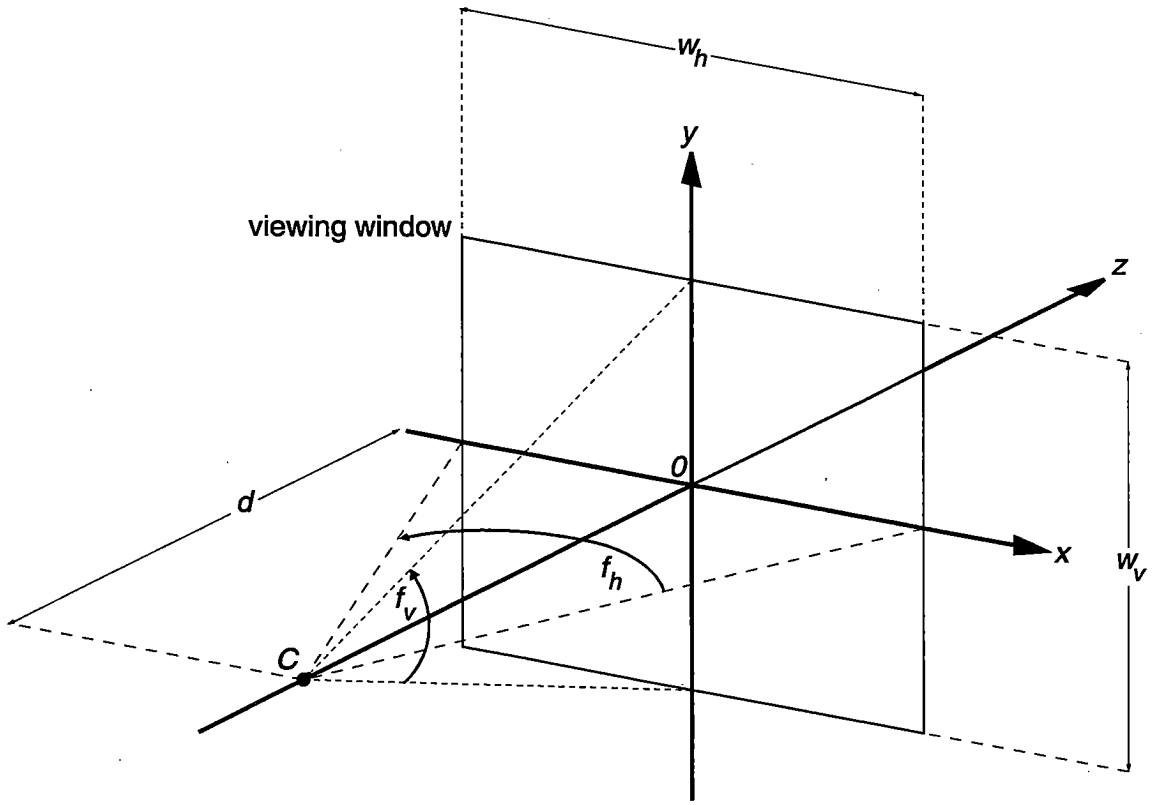


Figure 4.3: Monoscopic field-of-view

These dimensions are subsequently used to scale the coordinates of projected image points on the viewing plane to device coordinates. If the aspect ratio of a pixel on the display device is taken into account, then it is conventional for only the horizontal field-of-view to be specified, with the vertical field-of-view chosen so as to make equal image space steps in the horizontal and vertical directions equal in terms of physical device distances as well. As with all the perspective projection geometry described in this and the preceeding sections, note that the model is only accurate if the actual position of the observer in front of the display screen matches the assumed position of the camera in relation to the viewing window. If the observer is at any other position, the image seen will appear distorted.

Consider a typical workstation environment. Assuming a 19-inch monitor with a 4:3 aspect ratio, an observer sitting 50cm away from the screen perceives a horizontal field-of-view of approximately 42° at the monitor screen. An observer 100cm away perceives a 22° field-of-view. If the field-of-view used to generate a full-screen perspective image does not match this, then the image will appear distorted. However, this distortion is rarely considered to be a problem in practice as we have be-

come so accustomed to seeing it in virtually all man-made images, from television sets to cinema screens, and from illustrations in books to paintings hung on the wall.

Although it is not generally possible to modify the size of the display screen under control of the rendering system, it is possible to compensate for this apparent field-of-view distortion to some degree by applying an inverse distortion to the image prior to display. Wiseman [1990] describes a method whereby a "view angle correction" transformation is applied to the image to take into account the field-of-view at which the image is expected to appear to the viewer on the display screen, as well as the field-of-view used to synthesise the image. The resulting image then appears to the observer as it would if viewed with the correct field-of-view, but in general it cannot be displayed at the correct scale due to the physical constraints of the display device itself. If the observed field-of-view is greater than the image field-of-view, the apparent effect on the image is similar to that obtained using a wide angle camera lens; if the observed field-of-view is less than the image field-of-view, an effect similar to a telephoto lens is produced. Unfortunately, the non-linear nature of the compensatory distortion transformation makes it difficult to apply efficiently to projective rendering methods such as the Z-buffer, although it may be readily adapted for use with more flexible ray tracing approaches.

4.2 Existing stereoscopic 3D viewing models

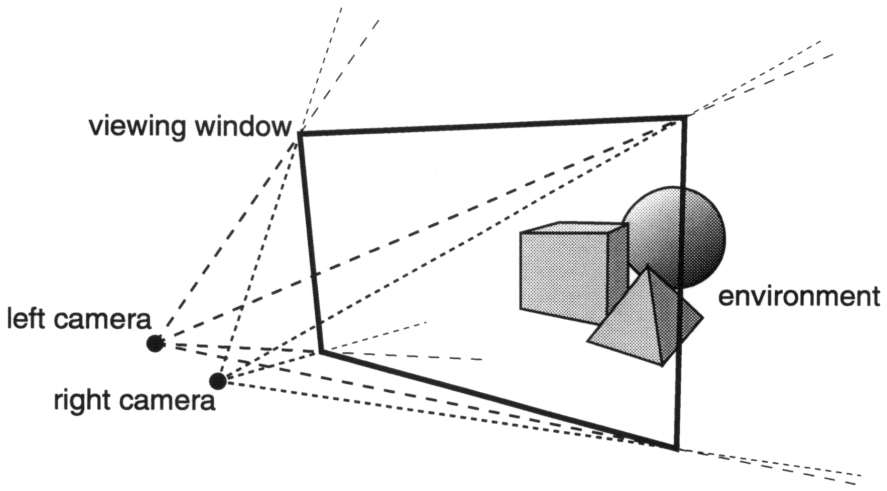
There are a number of stereoscopic 3D viewing and perspective projection models described in the literature, including those of Baker [1987], Tessman [1990], Williams and Parrish [1990], and Hodges [1991]. As noted by Hodges [1991], these models share much in common, with variations in terminology and formulation. The model described in this section is closest to that described in Hodges [1991] and Hodges [1992].

4.2.1 A stereoscopic camera viewing model

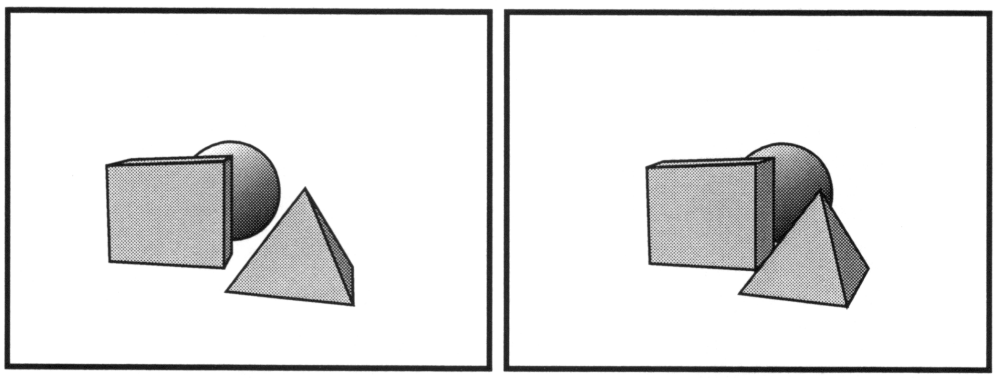
It is possible to describe a stereoscopic 3D viewing model as a logical extension of the monoscopic camera viewing model described in Section 4.1. Assuming a plano-stereoscopic display, the same virtual viewing window is used as before, representing the screen of the display itself. Where the monoscopic camera model used a single virtual camera to represent the observer's eye, a stereoscopic camera model uses two virtual cameras — one for each of the observer's eyes.

The positions of the two virtual cameras in relation to the viewing window should correspond with the positions of the observer's eyes in relation to the display screen. Assuming an upright observer directly in front of the centre of the

display, the camera viewpoints are located the same perpendicular distance away from the viewing window, centred vertically but offset horizontally from the centre of the viewing window. The viewpoint corresponding to the observer's left eye is offset to the left of centre of the viewing window and the viewpoint corresponding to the observer's right eye is offset to the right, such that the separation between the left and right viewpoints is compatible with the distance between the observer's eyes. This is shown in Figure 4.4.



(a) the relationship between the cameras, the viewing window and the environment



(b) what the observer sees on the display screen

Figure 4.4: Stereoscopic camera viewing model

The relative separation between the viewpoints is one of the most critical parameters in the production of high-quality stereoscopic 3D images: too great a separation will make the resulting pair of images uncomfortable or even impossible

to fuse stereoscopically, too small a separation will reduce or even eliminate any stereo depth information in the image. This issue is dealt with in Section 4.2.5.

If the actual position and orientation of the observer with respect to the display does not match the assumptions described above, distortions may be perceived in the stereo image. In particular, if the observer is looking at the screen from an off-centre position, the image will appear to be skewed about the screen plane. Combined with movement of the observer's head, this gives rise to an "apparent motions" effect, whereby objects appear to move and distort in response to viewer motion in a way which is not experienced in the real world, as noted by Harrison and McAllister [1993]. Worse still, if the observer's head is tilted to one side or the other, his eyes will be offset vertically as well as horizontally with respect to the stereo image and fusion may be difficult or even impossible to achieve. It is possible identify and compensate for these and other sources of distortion in the stereo image if head tracking is used (Deering [1992]) or if the display is head mounted (Robinett and Rolland [1991]), but only at considerable cost in the complexity of the viewing model.

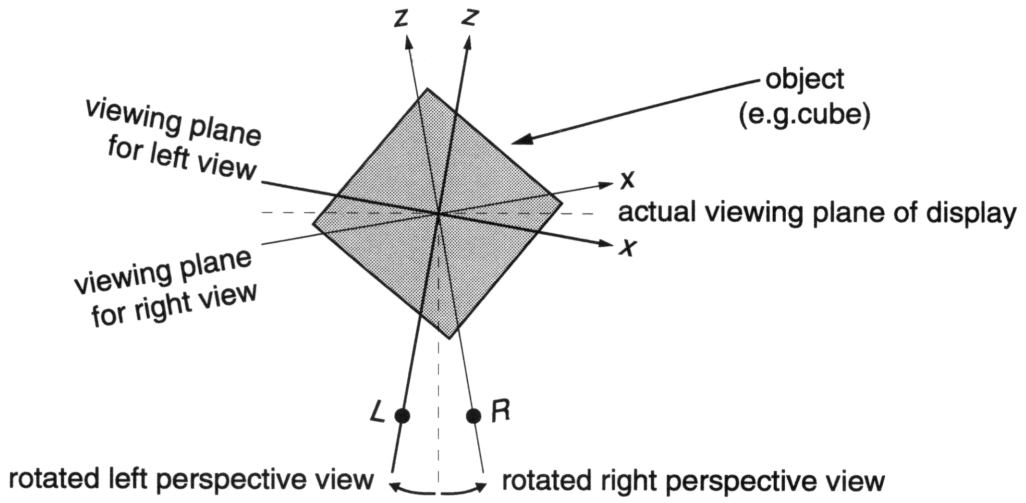
4.2.2 Reproducing the stereoscopic camera image

Reproducing a stereo image can be thought of as simply the problem of reproducing what would be visible from each of the left and right camera viewpoints as separate monoscopic views generated by the rendering process. The stereo display device then delivers these separate left and right camera views to the left and right eyes of the observer respectively, where the views are fused stereoscopically to form a coherent true 3D image.

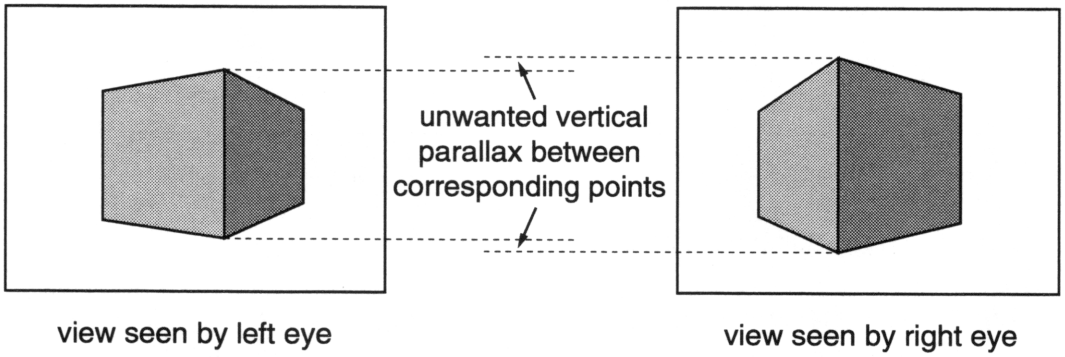
The rendering problem for each view of a stereo image is essentially the same as for monoscopic images, differing only in the position of the viewpoint relative to the viewing window. This similarity allows many of the techniques used for monoscopic image generation to be applied more-or-less directly to stereo images.

However, some care must be exercised when using monoscopic image generation techniques to produce stereoscopic images, in order to avoid introducing unwanted artifacts into the resulting image (Hodges [1991]). Perhaps the most common and widely-discussed stereo artifact in the literature occurs with views generated using separate perspective projections whose axes are rotated with respect to one another. The idea is to orient the views towards a central point in the scene in much the same way as each of our eyes rotate to converge at an object in the real world. A number of authors (Baker [1987]; Butts and McAllister [1988]; Lipscomb [1989]; Hodges [1991]) have noted that this approach introduces unwanted vertical parallax between corresponding points in the left and right images, as well as producing spatial distortions. An analysis of the geometry of the problem is presented

by Hodges and McAllister [1993] which quantifies the amount of vertical parallax in terms of the centre of rotation, the location of the view plane, and the angle of rotation. As Tessman [1990] points out, rotation before perspective projection effectively rotates the viewing plane for conventional *on-axis* perspective projection. With a plano-stereoscopic display, the rotated viewing plane for each view does not lie in the plane of the single fixed display surface on which the views are seen by the observer, as illustrated in Figure 4.5. The resulting vertical parallax and other distortions in the views can make the image difficult or even impossible to fuse stereoscopically, inducing eyestrain in the observer.



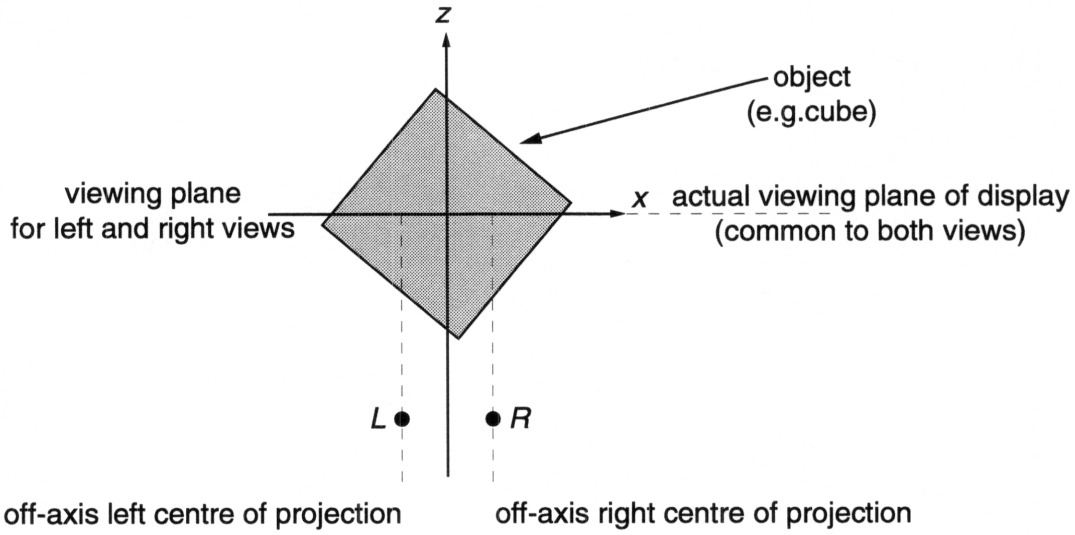
(a) geometry of rotation of perspective views for stereo



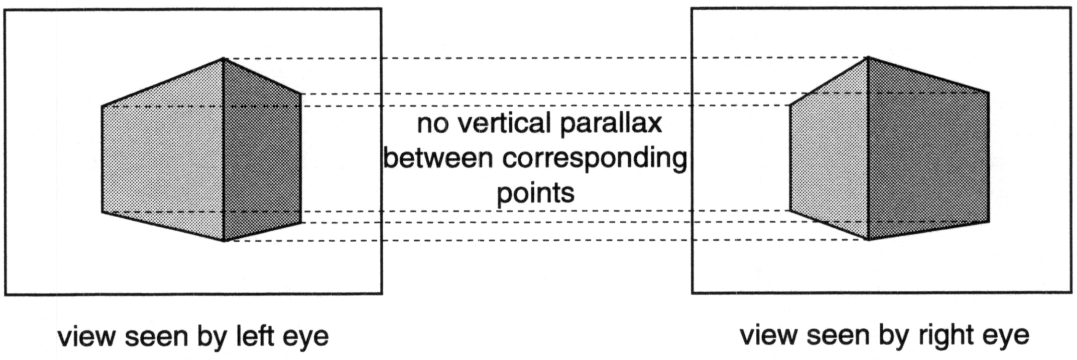
(b) distortions in stereo image due to rotation of perspective views

Figure 4.5: Stereoscopic artifacts resulting from rotation of on-axis perspective projections

One way to avoid these rotation artifacts with stereo perspective images is to use *off-axis* projections from the left and right viewpoints respectively. This approach eliminates any distortions due to unwanted vertical parallax between the views and is the method described in Section 4.2.3. Figure 4.6 illustrates how the off-axis projection avoids the artifacts produced by rotation of perspective views for stereo.



(a) off-axis perspective view geometry for stereo



(b) correct stereo image using off-axis perspective views

Figure 4.6: Off-axis stereoscopic perspective projection. Note how this avoids the artifacts produced by rotation of perspective views for stereo shown in Figure 4.5.

Orthographic projection in stereo suffers from a different problem. By its very definition, any two orthographic projections onto the same plane will produce identical images and thus will appear flat when viewed as a stereo pair. However it is

possible to produce a stereoscopic image using orthographic projection by rotating the projection planes relative to each other. As there is no perspective foreshortening effect, this approach does not introduce any unwanted vertical parallax. Unfortunately it does suffer from other spatial distortion problems, with objects appearing to grow in size the further away they appear, rather than diminishing in size as with perspective. This effect is the *anomalous perspective* illusion which occurs with orthographic projection in stereo, as noted by Lipscomb [1989] and Tessman [1990]. In addition, the viewable depth volume must be carefully constrained with rotated orthographic projections, as absolute stereoscopic parallax increases without limit with increasing depth (Hodges and McAllister [1993]). This can make it impossible to fuse the views stereoscopically and an uncomfortable double image effect results.

4.2.3 Stereo perspective projection geometry

The geometry of the perspective projection for each view of a stereo image is closely related to the geometry of a monoscopic perspective projection as described in Section 4.1.3. The principal difference is that the centre of projection for each view is offset horizontally from the centre of the viewing window, corresponding to the assumed position of each eye of the observer in front of the display screen. Thus the perspective projection used for each stereo view is an *off-axis* projection, as opposed to the *on-axis* projection commonly used for conventional monoscopic perspective. This technique and its variations have been described by several authors in the literature, including Baker [1987], Tessman [1990], Williams and Parrish [1990], and Hodges [1991].

The off-axis perspective projection geometry is shown in Figure 4.7. A left-handed viewing coordinate system is assumed as before. In contrast to the monoscopic perspective projection, there are two centres of projection — one for each of the camera viewpoints, corresponding to the left and right eyes of the observer. The viewpoints are located at the same perpendicular distance d from the viewing plane, evenly spaced horizontally about the centre of the viewing window. If the viewpoint separation distance is e, then the left and right centres of projection are $C_L(-e/2, 0, -d)$ and $C_R(+e/2, 0, -d)$ respectively. A point $P(X, Y, Z)$ in the scene is projected onto the viewing plane to point $Q_L(x_L, y_L, 0)$ in the left view and point $Q_R(x_R, y_R, 0)$ in the right view, where:

$$x_L = \frac{X \cdot d - Z \cdot e/2}{d + Z} \tag{4.5}$$

$$y_L = \frac{Y \cdot d}{d + Z} \tag{4.6}$$

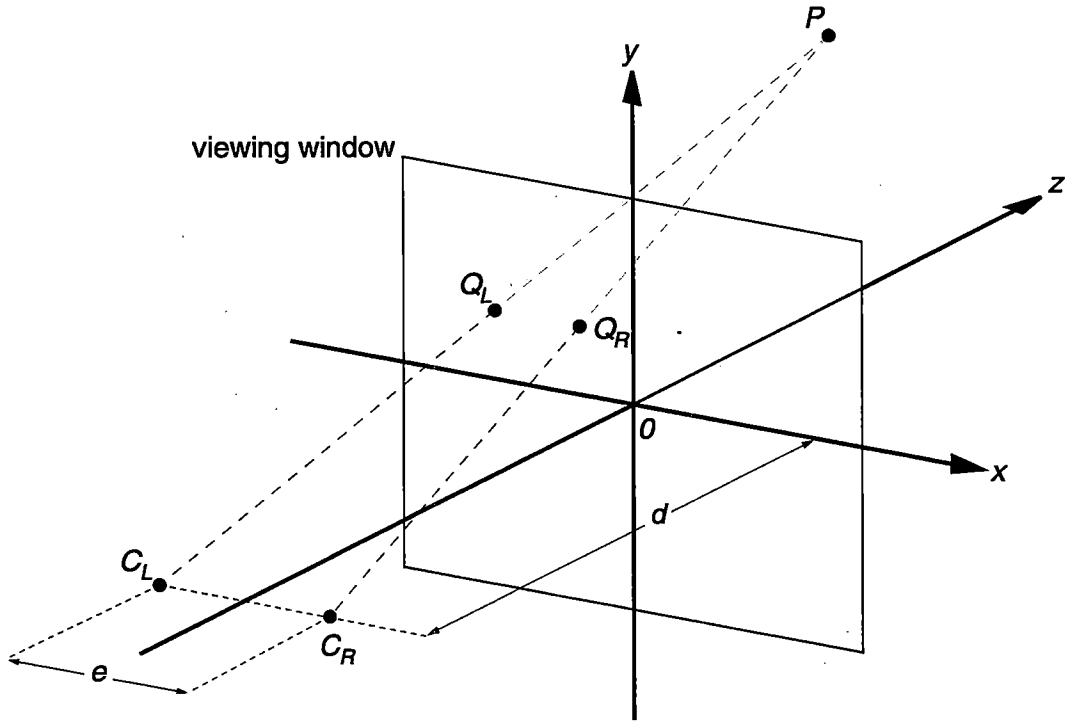


Figure 4.7: Stereoscopic perspective projection geometry

$$x_R = \frac{X \cdot d + Z \cdot e/2}{d + Z} \tag{4.7}$$

$$y_R = \frac{Y \cdot d}{d + Z} \tag{4.8}$$

Studying Equations 4.5 to 4.8, two important observations can be made. The first is that the expressions in Equations 4.5 and 4.7 for the horizontal components (x_L and x_R) of the projected points (Q_L and Q_R respectively) are very similar, differing only in the signs of the offset of the viewpoint from the centre ($e/2$). The second is that the expressions in Equations 4.6 and 4.8 for the vertical components (y_L and y_R) of the projected points are identical. This provides some scope for reducing the computation required to generate both left and right projections, as well as guaranteeing that there is no vertical parallax between corresponding points in the two views.

Alternative views of the scene may be obtained by moving the camera viewpoints and the viewing window relative to the scene. As with the monoscopic camera model, this may be achieved by applying the viewing transformation to each element in the scene. In the geometry described above, both left and right viewpoints share a common frame of reference which is in fact exactly the same as that

in the monoscopic camera model. As they share the same viewing transformation, the combined pair of stereo cameras may be moved and oriented in the environment much as if they were a single camera placed midway between the two stereo cameras. The only additional control parameter required to described the state of a stereoscopic camera model is the horizontal separation between the cameras which produces the disparity between the views seen by each camera.

4.2.4 Stereoscopic field-of-view

Increasing the size of the viewing window allows more of the scene to become visible to each of the stereo cameras, effectively increasing the field-of-view for each stereo camera in a similar manner to the monoscopic field-of-view as described in Section 4.1.4. As both stereo views must share the same viewing window, the image field-of-view scaling factors for each view must be the same in order to maintain perspective correspondence in stereo. It is therefore convenient to specify the stereo image field-of-view in the same manner as for a monoscopic image, with respect to an imaginary viewpoint centred between the left and right camera positions, and to apply the resulting field-of-view scaling factors to both views. However, the off-axis viewpoint positions of the stereo cameras produce asymmetric left and right fields-of-view which differ from the symmetric field-of-view of the conventional on-axis monoscopic viewpoint. This is illustrated in Figure 4.8, with the relative distance between the viewpoints exaggerated to demonstrate the effect.

A stereoscopic image can only be observed where the fields-of-view of both the left and right eyes intersect; a monoscopic view is seen by one eye otherwise. For regions behind the viewing plane this is perfectly acceptable as it agrees with the everyday experience of looking through a window frame, where objects beyond the window may be obscured to one eye but not the other. Monoscopic regions in front of the viewing plane do not correspond with real-life experience however and may cause some problems if any part of the environment falls into them. This is due to the fact that the brain does not know how to interpret objects which appear to be located stereoscopically in front of a window and yet which also appear to be occluded by its frame. This typically produces a considerable degree of perceptual stress for the viewer and may even result in a loss of apparent stereo in the image. For this reason, care must be taken when composing a stereo image to minimise the chance of this occurring. One way of achieving this is with the judicious application of clipping to remove those parts of the environment which lie in the affected regions.

Clipping is also used to identify any part of the environment which falls outside the field-of-view of the image and which therefore may be safely ignored by the renderer. With conventional on-axis projection, the field-of-view (and thus the

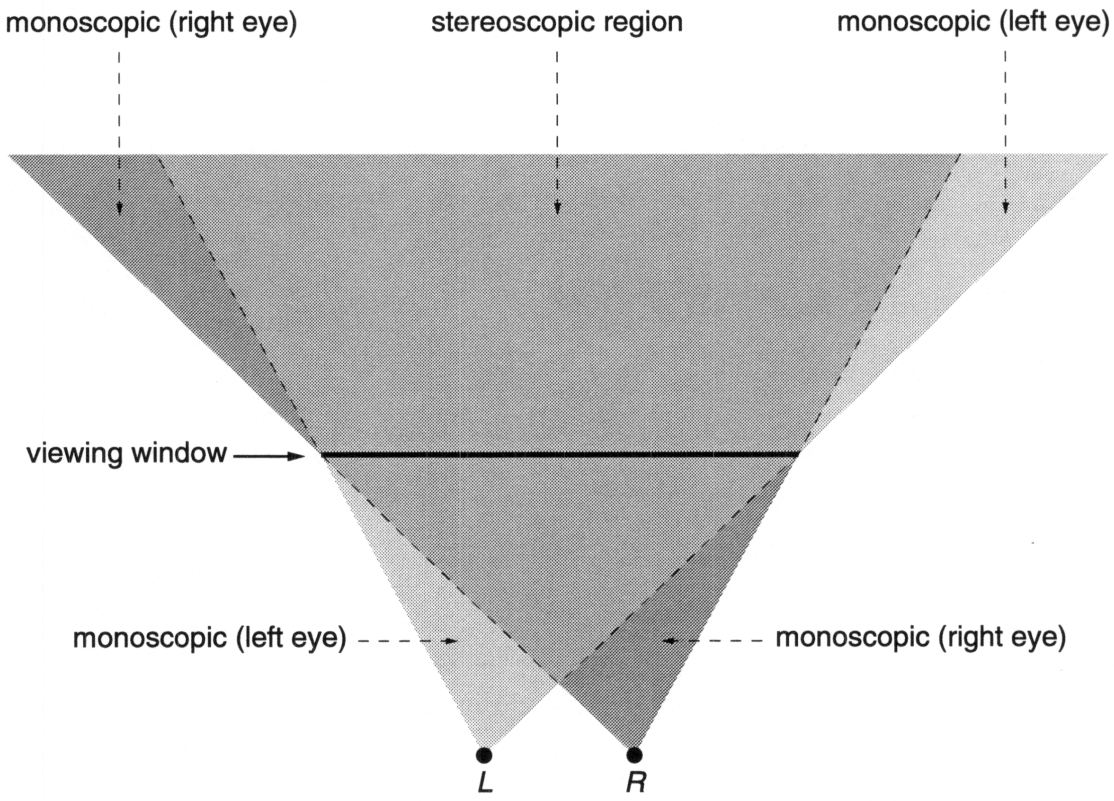


Figure 4.8: Asymmetric left and right fields-of-view in a stereo image. A stereo image can only be observed in the region corresponding to the intersection of the fields-of-view of both the left and right eyes. Note that while the monoscopic regions behind the viewing window are acceptable, any part of the environment which falls into the monoscopic regions in front of the viewing plane may induce considerable perceptual stress in the viewer and thus should be avoided.

clipping region) is symmetric about the projection axis. With the off-axis projection used for each view in a stereo image, the field-of-view is not symmetric as shown in Figure 4.8, and hence an asymmetric clipping region should be used by the renderer. Unfortunately many conventional renderers only support on-axis projection with a symmetric clipping region, which makes it difficult to use them to generate stereo images. However, it is possible to achieve an effect that is geometrically equivalent to off-axis projection by translating the centre of projection parallel to the viewing plane prior to on-axis projection, and then subsequently translating the projected image data back by the same amount in the opposite direction. As described in Williams and Parrish [1990] and Hodges [1991], this technique has some problems caused by the symmetric clipping normally used with a conventional on-axis projection. These problems include a reduction in the stereo field-of-view and

the potential loss of projected image data, as illustrated in Figures 4.9 and 4.10.

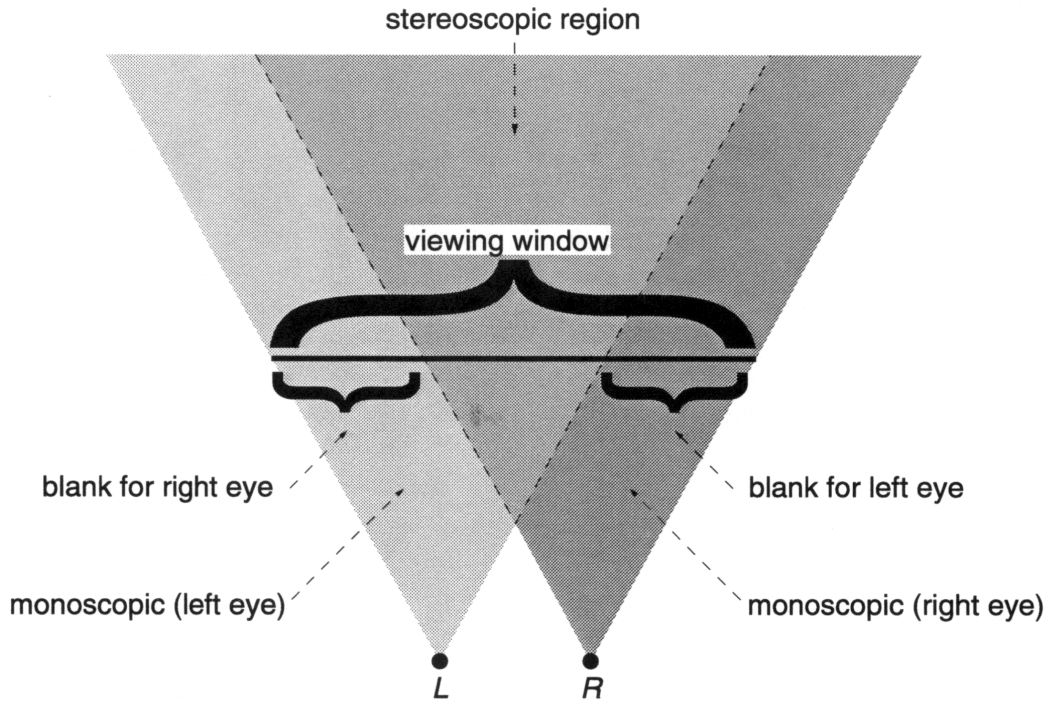


Figure 4.9: Conservative symmetric clipping for stereo images generated using on-axis projection. The field-of-view is contained within the viewing window for both the left and right views.

In Figure 4.9 the field-of-view is determined by the nearer of the left and right sides of the viewing window to each viewpoint, as described by Williams and Parrish [1990]. This suffers from relatively large blank areas in each view and thus a smaller central stereo region, but no loss of projected image data. The larger field-of-view used in Figure 4.10 is the same as that which would be used for conventional monoscopic on-axis projection with a viewing window of the same size, as described by Hodges [1991]. This results in smaller blank areas in each view and thus a larger central stereo region, but unfortunately some of the projected image data now falls outside the viewing window and must be discarded before display.

The proposed alternative approach is to use a field-of-view that is wide enough to encompass both the left and right edges of the viewing window in both views, as shown in Figure 4.11. This avoids the blank areas common to the methods described by Williams and Parrish [1990] and Hodges [1991], albeit at the cost of a greater loss of projected image data in each view than either of the other approaches. However it does produce a stereo image generated by on-axis projection with symmetric clipping which is indistinguishable from that obtained using off-axis projec-

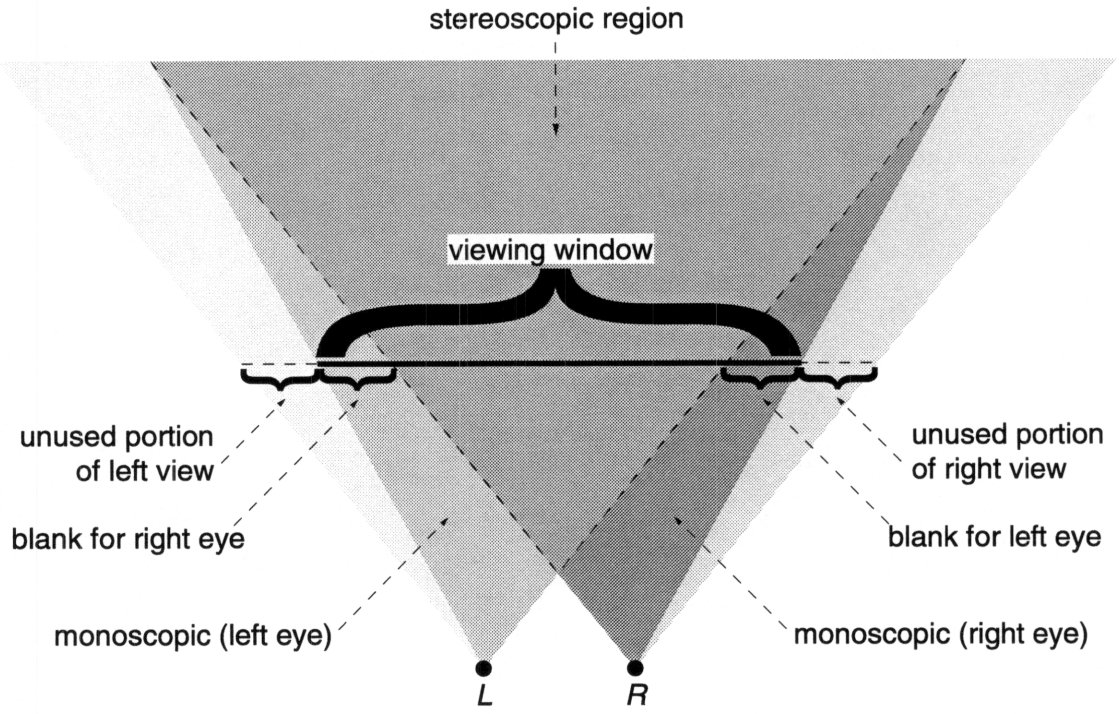


Figure 4.10: Conventional symmetric clipping for stereo images generated using on-axis projection. The field-of-view is the same as that used with conventional monoscopic viewing.

tion with asymmetric clipping, making it possible to use conventional rendering software (and hardware) with only a simple postprocessing image clip.

4.2.5 Stereo viewpoint separation

The selection of an appropriate level of disparity between the views is generally accepted as the most critical issue in the composition of stereoscopic images (Yeh [1993]). Too much disparity, and the views will be difficult for an observer to fuse into a single unified stereoscopic image, provoking viewer discomfort and the perception of double images. On the other hand, too little disparity may result in the image appearing somewhat flattened, with little or no noticeable stereo depth.

The relationship of the perceived depth of a point and the disparity between the views of that point in a stereo image is illustrated in Figure 4.12. There are two basic types of disparity: crossed and uncrossed. Crossed disparity (also known as *negative* disparity) is observed with points that lie on the near (viewer's) side of the display screen. Uncrossed disparity (also known as *positive* disparity) occurs with points that lie on the far side of the display screen. Points which lie precisely in the

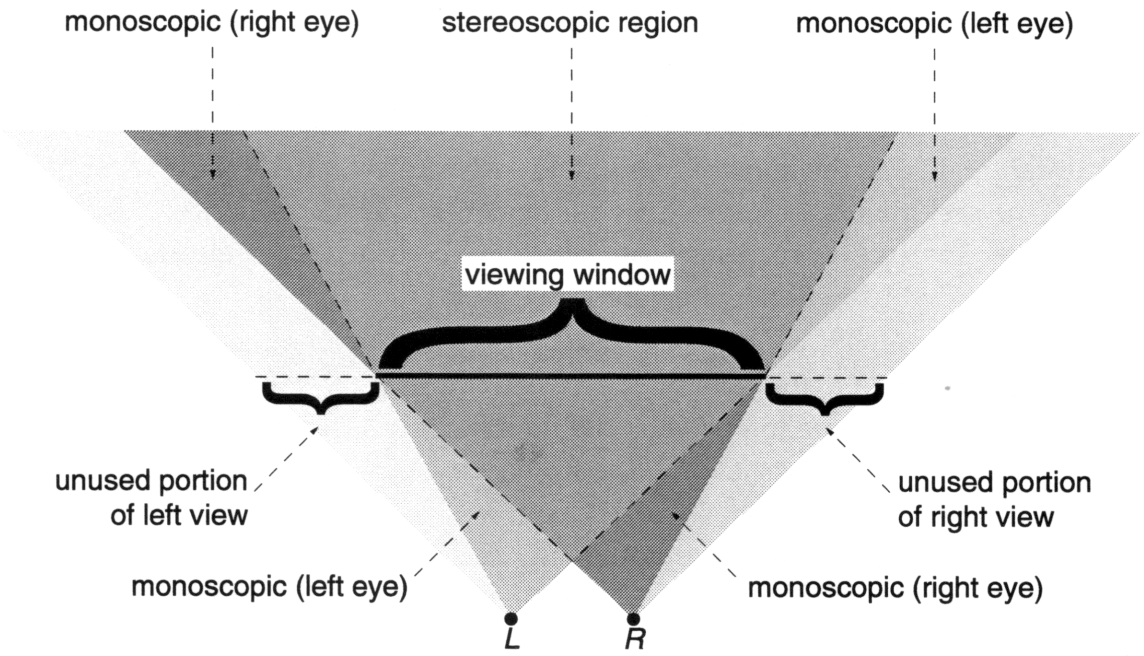


Figure 4.11: Expanded symmetric clipping for stereo images generated using on-axis projection. The asymmetric field-of-view shown in Figure 4.8 is a subset of the field-of-view shown here.

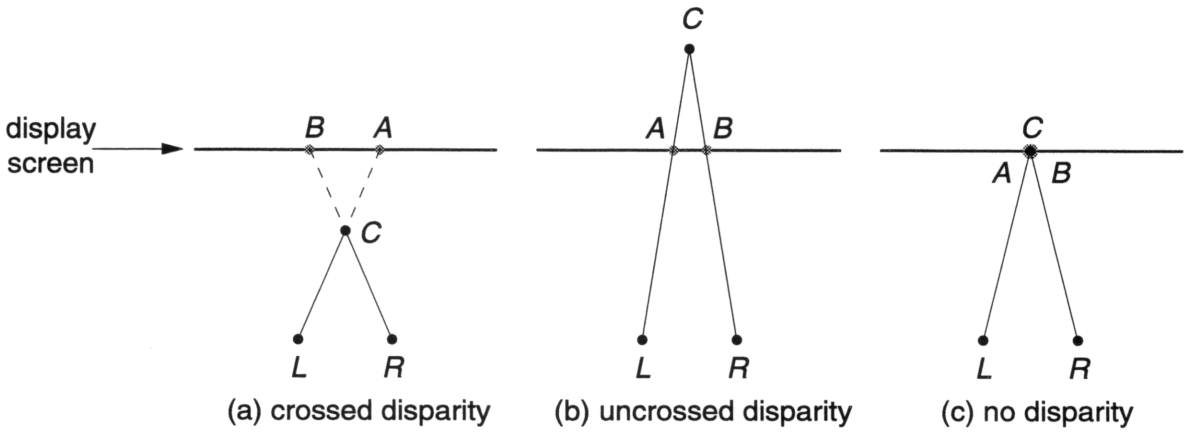


Figure 4.12: The relationship between stereo disparity and perceived depth. The image of point C is A in the view from L, and B in the view from R.

plane of the display screen have no disparity (also referred to as *zero* disparity).

In theory, it is possible for crossed disparity to have arbitrarily large negative values, as an object approaches ever closer to the observer. In practice however, negative disparity values are limited not only by the physical size of the display screen but also by considerations of viewer comfort. Uncrossed disparity is somewhat different, in that the maximum possible disparity is limited by the separation between the viewpoints, achieved only for objects an infinite distance away from the observer. In practice, such disparities may be too much for comfortable viewing, for reasons which will now be outlined.

The source of viewer discomfort with large disparity values is generally considered to be attributable to the difference between normal real-world viewing conditions and those which are imposed by the limitations of existing stereoscopic display devices. With natural viewing, the functions of accommodation (focussing) and convergence[2] of the eyes are closely coupled, so that when an observer fixates at a particular point, the eyes both focus and rotate to converge at the same point. With existing plano-stereoscopic displays however, this accommodation-convergence relationship is broken whenever the viewer turns his attention to a point at any depth other than that of the screen plane. This is because the observer's eyes are required to focus at the image on the screen plane, while simultaneously rotating to converge at the point some distance stereoscopically in front of or behind the screen (depending on the disparity of the point as seen in the left and right eye views). The effort required to maintain this decoupling of accommodation and convergence is thought to be the primary cause of the observer's perceptual strain and discomfort, particularly with prolonged exposure to images which exhibit large stereo disparities.

It is possible to vary the observed disparity for a point at a given depth from the screen by varying the distance between the stereo viewpoints. Determining precisely what constitutes an appropriate viewpoint separation may depend on a number of factors, including the distance of the observer to the screen, the size of the display screen, and the position of an object in relation to the view plane (Hodges [1991]). The intended scale of the observer in relation to the world space of the environment represented by the image should also be taken into account when selecting the viewpoint separation. For example, a stereo image of a city environment produced with a viewpoint separation corresponding to the width of several buildings would give the appearance of a table-top model (Tessman [1990]).

However, the viewer's comfort must also be considered in order to produce effective stereo images. Hodges [1991] claims that ignoring these factors can result in unfusable images with objects at large distances behind the screen, where the ob-

[2]sometimes referred to simply as *vergence*

served disparity approaches the eye separation of the viewer. The suggested rule-of-thumb is to use a viewpoint separation which allows a maximum disparity of no more than 1.5°. Yeh [1993] claims the fusion limit is in fact much lower if vergence response is eliminated by brief (200 ms) exposure to the stereo stimulus, with fusion limits of 27′ of arc for crossed disparity and 24′ of arc for uncrossed disparity. Fusion limits for more prolonged exposure (2s stimulus duration) are 4.93° for crossed disparity and 1.57° for uncrossed disparity, where the greater time allowed for vergence responses permits a much greater depth range to be fused. The results of another empirical study by Williams and Parrish [1990] suggest that the stereo depth volume of a display should be limited to lie between 25% of the viewing distance in front of the display screen and 60% of the viewing distance behind the screen.

Having determined a comfortable maximum range of permissable disparity, there is then the question of how to keep all objects in the image within these limits. One approach is simply to clip objects to suitable depth bounds, but this may not always be acceptable, particularly in an animated sequence. Another approach to the problem is to adjust the stereoscopic viewing parameters dynamically according to the content of the image in order to satisfy the disparity constraints. This is the basis of the methods described by McAllister [1992b] and Akka [1992]. However this continuous manipulation of viewing parameters may have a detrimental effect on the spatial perception of the observer and result in unreliable depth observations, as noted by Milgram and Krüger [1992].

As a final point on the representation of depth, it is worth noting that the separation of the viewpoints is by no means the only influence on depth perception and spatial relationships in a stereo image. Lipton [1993] claims that extrastereoscopic depth cues — perspective in particular — may play an important role in a stereo image by weighting or scaling the stereo cues. He suggests the use of exaggerated wide-angle perspective to enhance the perceived depth effect while maintaining comfortable stereo viewing, without the excessive disparities introduced by an exaggerated viewpoint separation. This is given some support by Yeh [1993], where it is noted that a viewer may experience anomalous perceptual effects in the absence of perspective cues in a stereo image. These effects include the apparent size of objects increasing with stereo depth, as predicted by the size-distance invariance hypothesis known as Emmert's law.

4.3 Multiple-viewpoint autostereo viewing model

Multiple-view autostereo displays are relatively rare compared to two-view stereo displays, with only a handful of such devices reported in the literature (Lang et al. [1992]; Isono et al. [1992]; Eichenlaub [1994]). Although these displays vary con-

siderably in their design and construction, they all share certain characteristics in common with their two-view counterparts: the 3D image produced only exhibits horizontal parallax between the views, and the same planar display format is used. This naturally leads to a logical extension of the viewing models described in Sections 4.1 and 4.2 for use with multi-view autostereo images[3].

4.3.1 An autostereo camera viewing model

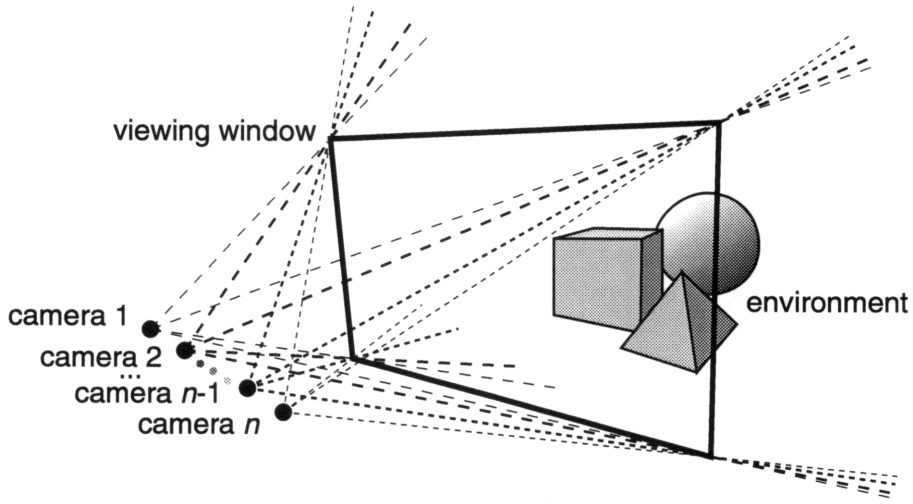
The multi-viewpoint autostereo camera viewing model described here is an extension of the two-view stereo camera model described in Section 4.2.1. The autostereo display is assumed to have a single planar viewing screen, which is represented by the virtual viewing window as before. Where the two-view stereo camera model uses only two virtual cameras to represent the observer's eyes, the viewing model for the n-view autostereo display uses n virtual cameras, representing the number of available viewing zones in front of the display.

The locations of the n virtual cameras in relation to the viewing window should correspond with the locations of the n available viewing zones in front of the display screen where each view is visible from. Assuming an upright observer situated in the field-of-view of the display, the camera viewpoints are positioned the same perpendicular distance from the viewing window, centred vertically but offset horizontally from the centre of the viewing window. Assuming equal-sized viewing zones in front of the display, the n viewpoints are distributed evenly across the field-of-view of the display, such that the separation between a given pair of viewpoints is compatible with the separation of the observer's eyes at the expected viewing distance. This is illustrated in Figure 4.13.

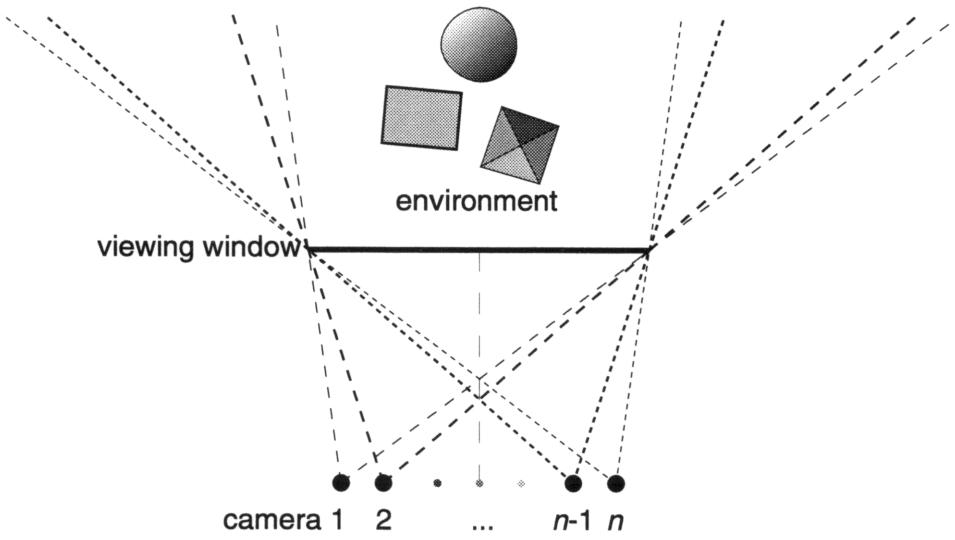
The relative separation of the viewpoints is as critical to a high-quality result in a multi-view autostereo image as it is in a two-view stereo image. However, the calculation of a suitable separation is complicated for a multi-view autostereo image by the fact that the relationship of the observer's eyes to the viewing zones may vary with the number of viewing zones available and the size of the viewing zones in relation to the distance between the observer's eyes. This issue is discussed in more detail in Section 4.3.5.

In contrast to the two-view stereo case, the observer has more freedom to move in front of the display with less distortion in the perceived image. In particular, movement of the observer's head from side to side allows alternative views to be seen from the corresponding viewpoints — this is the "look-around" effect. As each view is projected through a single viewpoint, observation from non-aligned view-

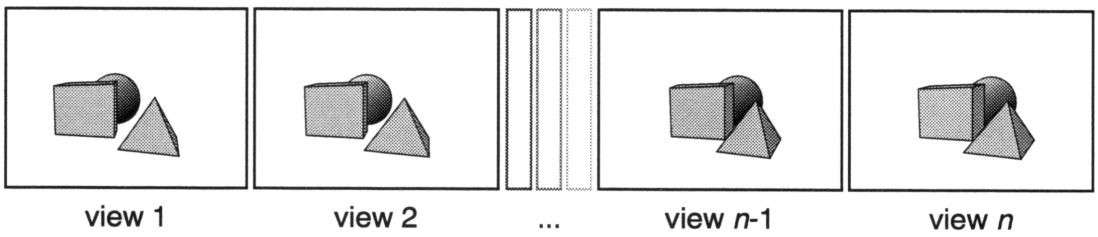
[3]note that the similarities between multi-view autostereo images and horizontal parallax only (HPO) holographic stereograms (Benton [1982]; Halle [1994]) means that much of the discussion which follows applies equally well to both.



(a) the relationship between each camera, the viewing window and the environment



(b) plan view of (a)



(c) what the observer sees in each view on the display screen

Figure 4.13: Autostereo camera viewing model

ing positions may still produce spatial distortions and apparent motion effects, but the smaller the spacing between the viewpoints the less noticeable these artifacts become. However, vertical movement of the observer's head still produces similar apparent motion effects, and tilting the head with respect to the display still hampers stereoscopic fusion. Theoretically, a display which used both vertical as well as horizontal parallax views would be capable of addressing these problems, but only at a correspondingly much higher cost in terms of bandwidth, or (alternatively) at a much lower resolution in each view.

4.3.2 Reproducing the autostereo image

The process of reproducing a multi-view autostereo image is essentially the same as reproducing a two-view stereo image: the concepts involved are very similar, with many of the techniques for two-view stereo extending directly to the multi-view case. Thus an n-view autostereo image can be produced by determining what would be visible from each of the n camera viewpoints in turn, as separate monoscopic views generated by the rendering process. The autostereo display hardware makes each of the n views visible only in the appropriate direction, and each eye of an observer looking at the display sees a different view corresponding to its position in relation to the display. These views are then fused stereoscopically in the normal way and the observer perceives a 3D image.

As for two-view stereo images, autostereo images may suffer from unwanted artifacts if care is not taken when applying conventional monoscopic image generation techniques to the multi-view case. In particular, off-axis perspective projection of the n views is used in a manner similar to that outlined in Section 4.2.2 to match the logical projection geometry to the physical geometry of the display device. The geometry for multi-view autostereo perspective projection is described in Section 4.3.3.

4.3.3 Autostereo perspective projection geometry

The approach described in this section extends the stereo perspective geometry from Section 4.2.3, generalising the use of off-axis projections from two to n views. Each of the n views has its own centre of projection, corresponding to the position in front of the display from which the view is assumed to be visible.

The geometry of multi-view autostereo perspective projection is illustrated in Figure 4.14. A left-handed viewing coordinate system is assumed. Instead of the two centres of projection (left and right) used in two-view stereo, there are now n centres of projection $C_1, C_2, \ldots, C_n$ each corresponding to one of the n viewpoints. The viewpoints are all located the same perpendicular distance d from the viewing

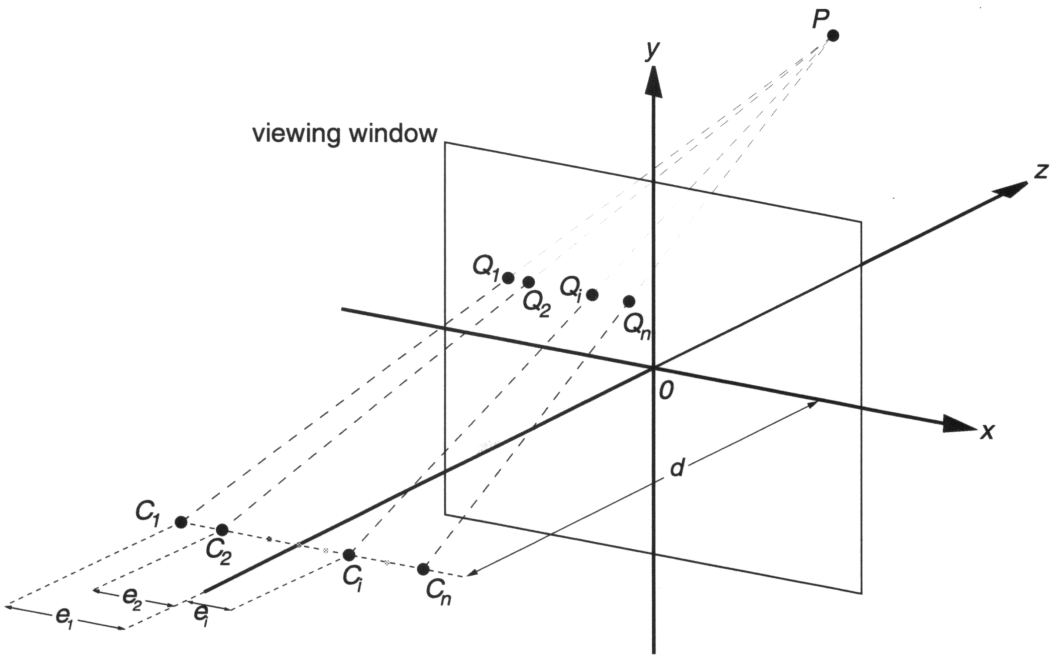


Figure 4.14: Autostereo perspective projection geometry

plane, distributed horizontally about the centre of the viewing window. If the horizontal offset for viewpoint i is e_i, then the coordinates of the centre of projection for view i are $C_i(e_i, 0, -d)$. A point $P(X, Y, Z)$ in the scene is projected onto the viewing plane in view i at point $Q_i(x_i, y_i, 0)$, where:

$$x_i = \frac{X \cdot d + Z \cdot e_i}{d + Z} \tag{4.9}$$

$$y_i = \frac{Y \cdot d}{d + Z} \tag{4.10}$$

Similar observations may be made about Equations 4.9 and 4.10 as were made about Equations 4.5 to 4.8 on page 70. The expression in Equation 4.9 for the horizontal component (x_i) of a projected point (Q_i) in view i depends only on the value of the horizontal offset (e_i) for that view, given a particular scene point P and view plane distance d. In Equation 4.10 the expression for the vertical component (y_i) of the projected point is independent of the view it appears in, thus eliminating the possibility of unwanted vertical parallax among corresponding points in the autostereo views. How these characteristics may be utilised to reduce the computation required for multi-view perspective image generation is discussed in Section 5.8.

Alternative views of the scene may be obtained by moving the autostereo camera viewpoints and the viewing window relative to the scene in the same manner as the two-view stereo camera model as described in Section 4.2.3. The viewing transformation applied to each element in the scene is the same for all views and thus it is possible to treat the autostereo cameras as if they were a single monoscopic camera positioned at the centre of the set of n autostereo cameras.

4.3.4 Autostereo field-of-view

The field-of-view for a multi-view autostereo image can be thought of as a simple extension of the field-of-view described in Section 4.2.4 for a two-view stereo image. The apparent field-of-view for a multi-view autostereo image may be increased by increasing the size of the viewing window as in Section 4.1.4. The off-axis viewpoint positions in a multi-view autostereo image result in a different asymmetric field-of-view for each view. This is shown in Figure 4.15 for a four-view image for the sake of clarity, although the same principles apply for images with a greater number of views.

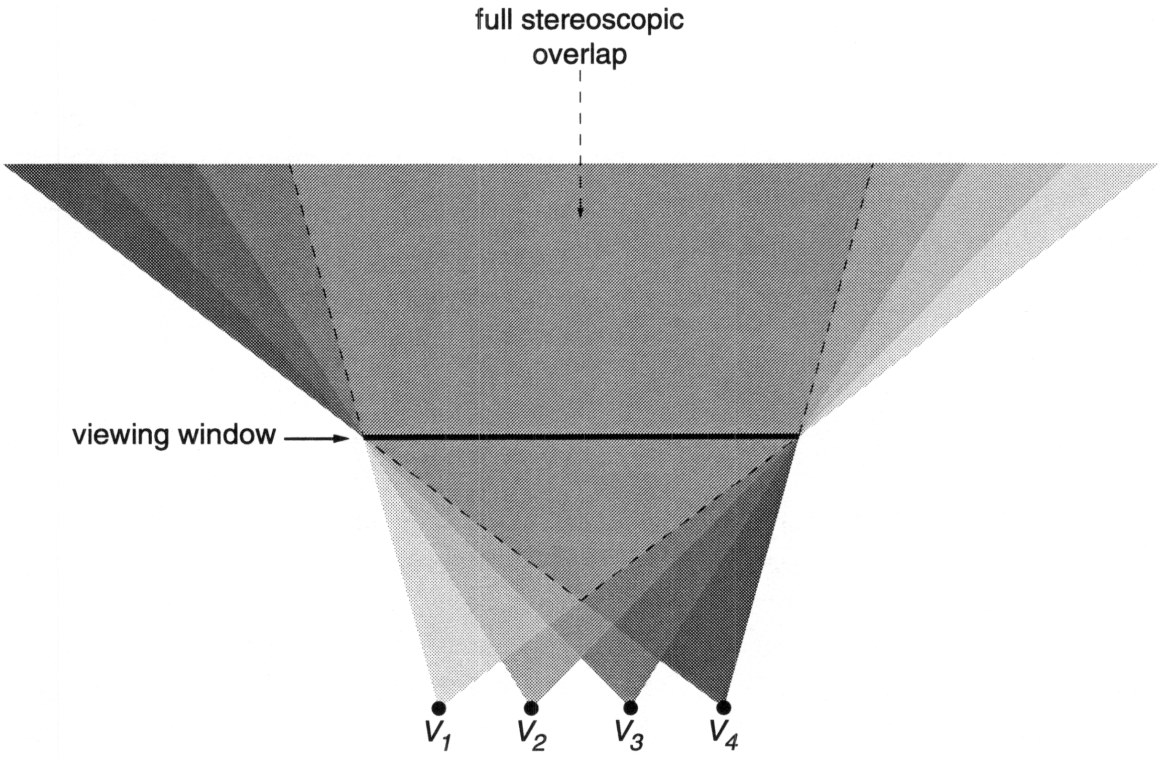


Figure 4.15: Multi-view autostereo field-of-view

Although the fields-of-view for each view of a multi-view autostereo image overlap in a much more complex manner than in a two-view stereo image, the principles of where stereo can and cannot be seen are essentially the same. The main difference for multi-view autostereo is that the question of whether a point in the environment will or will not appear in stereo now depends on the positions of the observer's eyes in relation to the viewing window, because different parts of the environment may be seen from different locations. In general, only objects which fall into the large central region where the fields-of-view for all the views overlap can always be seen in stereo.

If off-axis projection is used, the asymmetric fields-of-view in a multi-view autostereo image will have to be taken into consideration by the renderer when clipping. It is still possible to use a combination of on-axis projection and translation to produce each stereo view with conventional symmetric clipping as in Section 4.2.4, but the cost of the technique may be much higher than for the two-view case. This is because the maximum offset of each viewpoint from the centre is generally greater for multi-view stereo images, so the maximum field-of-view required to span the full width of the viewing window from a given viewpoint will be correspondingly greater. This in turn will lead to even larger unused portions of each view, which may result in a significant loss of efficiency and consequent rendering performance.

4.3.5 Autostereo viewpoint separation

For two-view stereo images, the viewpoint separation can be treated as a quantity analogous to the actual distance between the eyes of the observer. For a multi-view autostereo image however, the relationship between viewpoint separation and the observer's interocular distance is not as straightforward, as it depends on a variety of factors related to the physical viewing geometry of the display.

Consider the physical viewing geometry shown in Figure 4.16. An observer with interocular distance ΔE is looking at the display from a distance D. For convenience, the observer's eyes are assumed to be the same distance from the viewing plane as the cameras, that is $D = d$. The display produces an image made up of n views visible over an angular field-of-view θ. At the viewing distance D the n views span a field-of-view[4] of width W, where:

$$W = 2D \tan \frac{\theta}{2}$$

[4]Note that this physical display field-of-view is not in any way related to the autostereo image field-of-view discussed in Section 4.3.4.

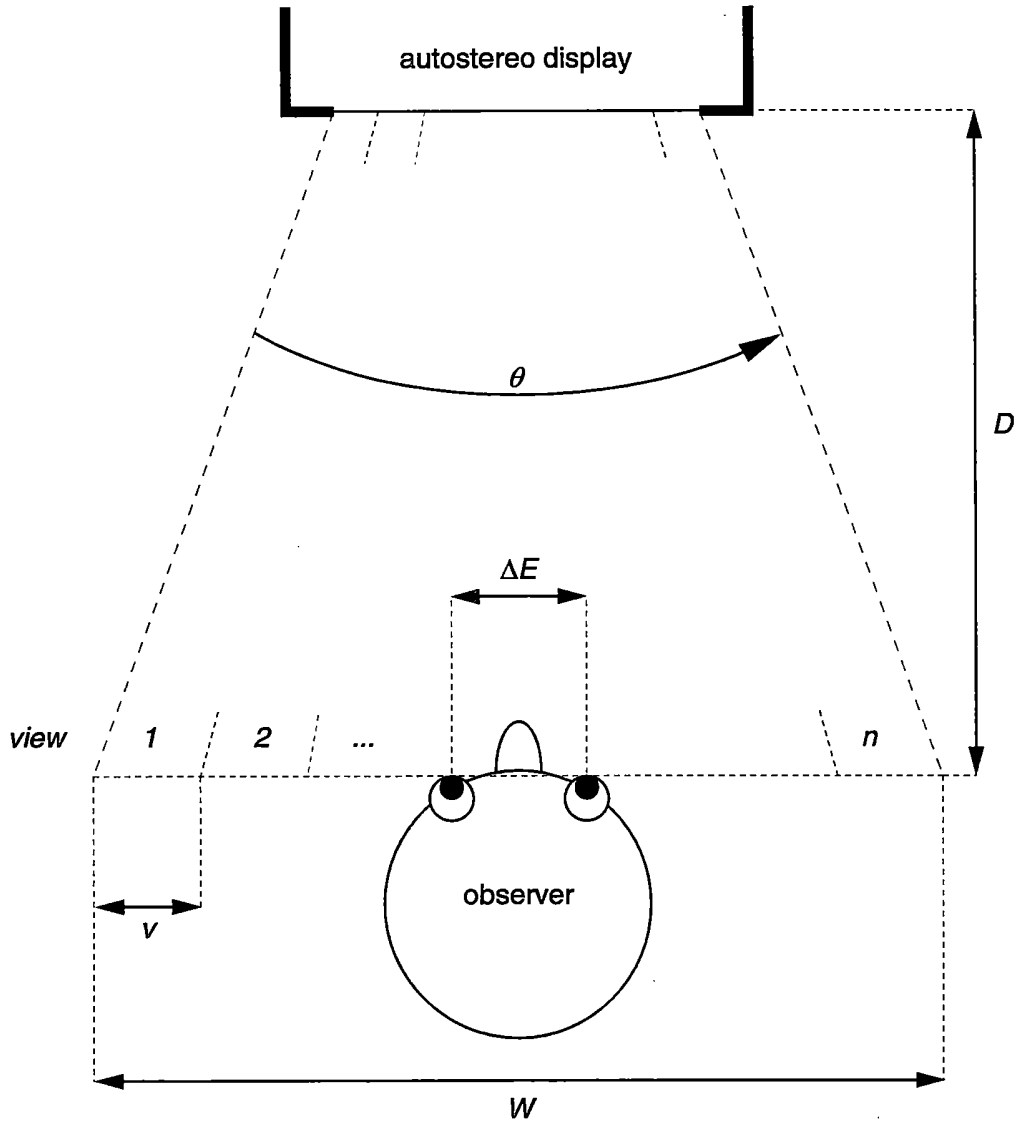


Figure 4.16: Physical viewing geometry of the autostereo display

Assuming the n views are distributed evenly in space, each of the views is visible over a width v at the specified viewing distance d, where:

$$v = W/n \tag{4.11}$$

If the camera viewpoint for each view is positioned in the middle of its respective viewing zone, the distance from one viewpoint to the next is v. Although the actual separation between the views seen by the left and right eyes of the observer depends on the exact position of the observer with respect to the viewing zones, the expected average separation $\bar{s}$ between the left and right eyes in terms of number of views can be expressed as

$$\bar{s} = \Delta E/v$$

If $\bar{s} < 1$ it is possible that both of the observer's eyes may see the same view, and thus no stereo depth may be perceived. If $\bar{s} \geqslant 1$ each of the observer's eyes will always see a different view, thus allowing stereopsis to occur.

In general $\bar{s}$ may not be an integer, and therefore it may be possible for the observer to perceive different amounts of stereo separation between the left and right eye views depending on his position with respect to the viewing zones. For example, Figure 4.17 illustrates an example where $\bar{s} = 1.5$. An observer at position A sees view i with the left eye and view $i+1$ with the right. The same observer at position B still sees view i with the left eye, but now sees view $i + 2$ with the right. This makes the view separation between the eyes change from 1 to 2 views in moving from A to B, a doubling of the stereo separation which may induce a pronounced shift in apparent depth. The same observer at position C now sees view $i + 1$ with the left eye, but still sees view $i + 2$ with the right, thus reducing the view separation to 1 view again. A similar pattern repeats itself across the entire field of view, resulting in the image appearing to "wobble" depth-wise in and out of the screen as the observer moves from side to side, in a somewhat distracting manner.

In general, for an average view separation $\bar{s}$ between the left and right eyes of the observer, the perceived view separation alternates between $\lfloor \bar{s} \rfloor$ and $\lceil \bar{s} \rceil$ views as the observer moves from side to side[5]. As the apparent depth of the image depends on the perceived view separation, there will be a variation in the depth seen by an observer depending on his position in relation to the display. The relative magnitude of this depth variation depends on the relative magnitude of the variation in perceived view separation, thus the effect is most noticeable for small values of $\bar{s}$.

[5] $\lfloor x \rfloor$ is the greatest integer less than or equal to x; $\lceil x \rceil$ is the smallest integer greater than or equal to x.

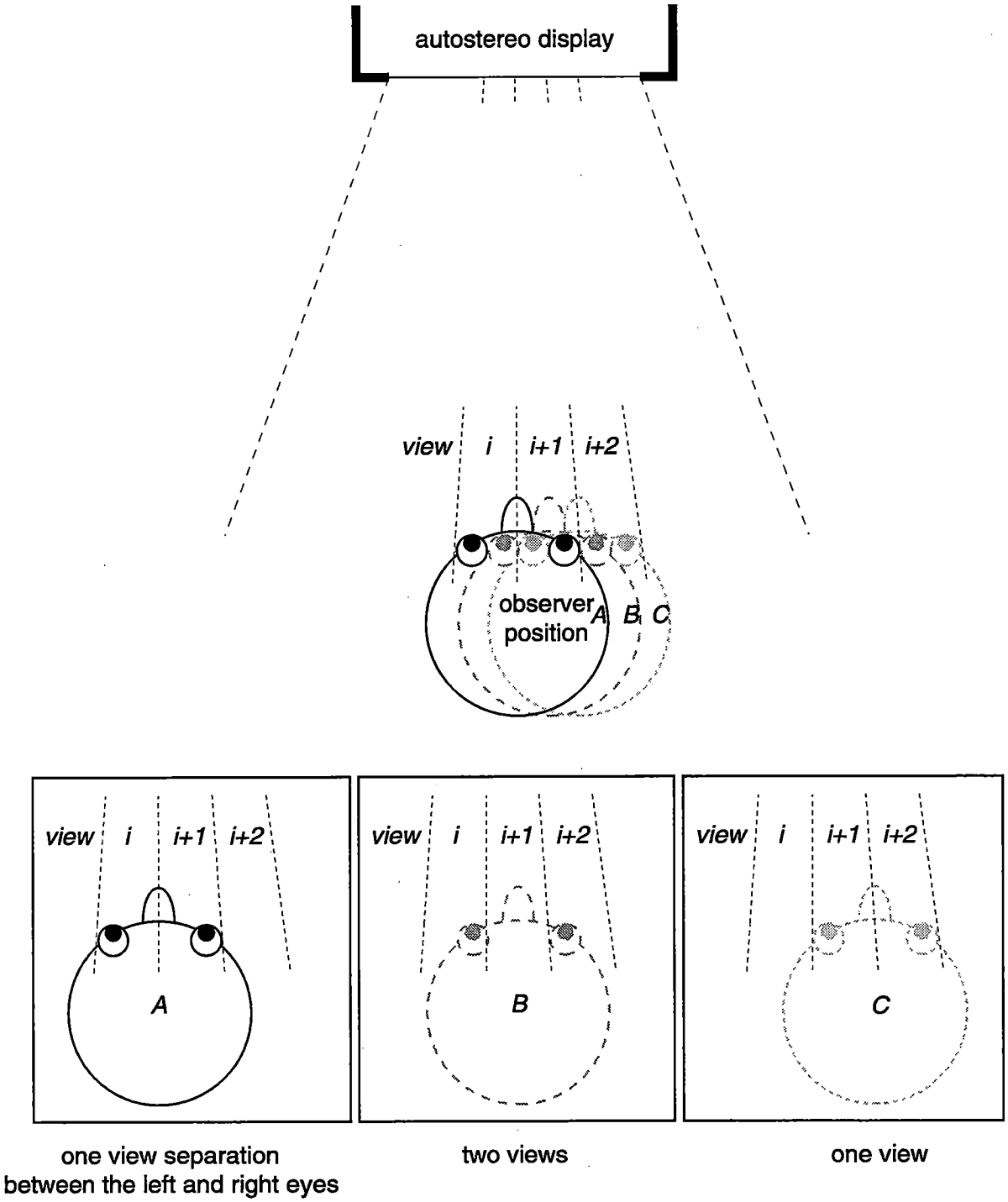


Figure 4.17: "Depth wobble" in an autostereo image. The apparent view separation (and thus depth) may vary with viewing position when the width of each viewing zone is not an integer multiple of the interocular distance.

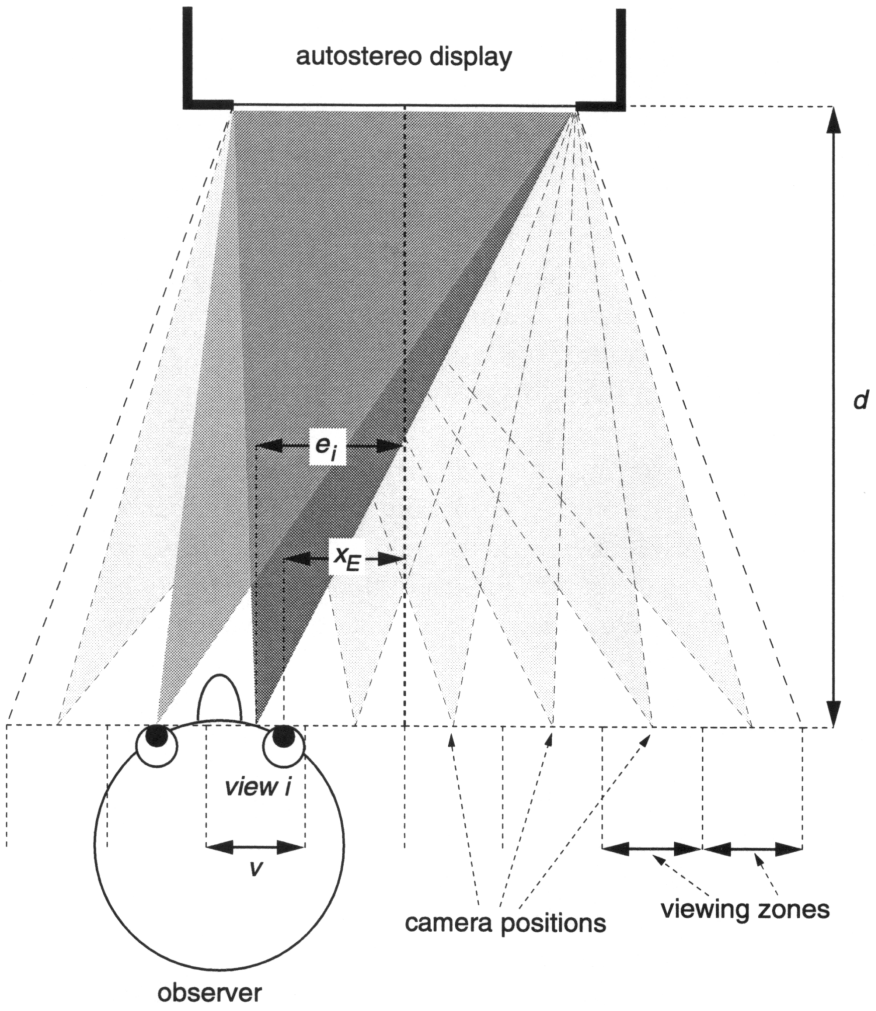


Figure 4.18: Autostereo viewing position error

This effect can also be thought of in terms of the error between the ideal view that corresponds what should be seen from the position of each eye of the observer and the view actually made visible to each eye by the display hardware. This is a direct consequence of the fact that there are only a discrete set of views displayed, yet each view is visible from a continuous region of space. In Figure 4.18, the position of the observer's eyes are shown in relation to the assumed positions of the virtual cameras in the centre of each viewing zone. For an eye at position $E(x_E, 0, -d)$ to see view i it must be offset horizontally from the corresponding camera position e_i by less than half the width of the viewing zone, thus:

$$|x_E - e_i| \leqslant v/2 \tag{4.12}$$

From Equation 4.12, it can be seen that it is possible to control the maximum possible error in viewing position by changing the width v of each viewing zone. As v is decreased, so the maximum viewing position error is reduced and the effect of the variation in perceived stereo disparity is attenuated. However Equation 4.11 implies that decreasing v requires trade-offs in other areas. One possibility is to reduce the overall field of view of the display, but this may place undesirable restrictions which this places on the permissible range of the observer's head movement. Alternatively, the number of views could be increased, but the increase in image bandwidth required to support these additional views places correspondingly greater demands on both the display hardware and the image synthesis system.

4.4 Summary

In this chapter, the viewing model used to guide the synthetic image generation process has been examined in some detail. A multi-view autostereo viewing model has been developed, based on an extension of existing conventional monoscopic and two-view stereoscopic viewing models. Armed with this knowledge and understanding of the geometrical relationships between the views in a multi-view autostereo image, it is possible to explore how the similarities between the views may be taken advantage of by a multi-view stereo renderer to improve rendering efficiency. Chapter 5 describes how these ideas may be applied to a multi-view stereo renderer based on a Z-buffer visible surface algorithm.

5

The Stereo Z-buffer

This chapter describes some of the issues involved in the design and implementation of a multi-view stereo rendering system based on a Z-buffer approach to visible surface determination. Imaginatively dubbed the Stereo Z-buffer, the algorithm takes advantage of geometric coherence between the stereo views to improve rendering performance at a number of levels by sharing information among all views in the stereo image. First, an overview of the algorithm is given and a model of the stereo rendering pipeline is presented. This pipeline is then used as the framework for the discussion of how stereo coherence can be employed to improve performance at each stage in the rendering process. Finally, a brief outline of how the system may be adapted to incorporate multi-pass antialiasing is given, along with a novel application of this technique to synthetic multi-view autostereo images.

5.1 Previous work

It is common practice for most computer-generated stereo images to be obtained by treating each view as a separate image, with conventional monoscopic rendering techniques being applied independently to each (Hodges [1992]). Although this is conceptually simple, in practice there may be some problems applying conventional techniques (such as on-axis projection and symmetric clipping) to generate each stereo view, as discussed in Sections 4.2.4 and 4.3.4. Furthermore, this approach does not take into account any of the similarities between the stereo views

and the potential opportunities for gains in rendering efficiency they present.

Recently however there have been a number of rendering algorithms reported which have been specifically developed for generating stereo images. Most of the algorithms described are based on ray tracing techniques (Ezell and Hodges [1990]; Devarajan and McAllister [1991]; Adelson and Hodges [1993]). They use visible surface information from one view to infer an approximation of the other view in a stereo pair using reprojection. These approaches are reviewed in more detail in Section 6.1.1, together with related work by Guo et al. [1988], Chen and Williams [1993], and Ward [1994].

Papathomas et al. [1987] describe a method of using stereo coherence to optimise computation of stereo projection geometry for meteorology data displayed in a point cloud format. Adelson et al. [1991] describe a number of adaptations of conventional monoscopic techniques to a stereo context, including polygon scan conversion, line and polygon clipping, back-face removal, and visible surface determination using a hybrid BSP tree/Z-buffer approach as well as a stereo implementation of a scanline algorithm. At the time of writing, this latter paper appears to be the only reported work on a generic polygon rendering algorithm for stereo images which makes use of the coherence between the views to increase computational efficiency.

5.2 Outline of the algorithm

The multi-view stereo visible surface determination algorithm described here is based on a simple extension of a conventional single-view Z-buffer algorithm as outlined in Section 3.2.2.2. A model of the rendering process for a conventional single-view Z-buffer is shown in Figure 5.1, based on the rendering pipeline presented in Foley et al. [1990], section 16.14.

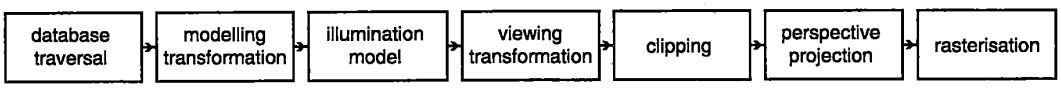


Figure 5.1: A model of the Z-buffer rendering pipeline

A brief description of each of these stages is given below.

database traversal converts the scene description into rendering primitives suitable for processing by the visible surface determination algorithm.

modelling transformation used to set the position, orientation and scale of each object relative to all others in the scene.

illumination model used to compute shading information for each primitive.

viewing transformation converts the world coordinate position of each point in the database to the camera frame of reference used for the subsequent perspective projection calculations.

clipping applied to each element in the database to allow the visible surface algorithm to only consider those elements (or parts thereof) which fall inside the viewing frustum of the camera and thus may be potentially visible.

perspective projection determines where each point in the database appears on the viewing plane.

rasterisation converts each graphical primitive to its corresponding pixel representation in the image, including the stages of scan conversion, visible surface determination using the Z-buffer, and shading.

For a stereo Z-buffer, each view in the stereo image has its own independent Z-buffer, with each pixel in a given view having a corresponding entry in the Z-buffer for that view. Visible surface determination *at the pixel level* is achieved in precisely the same manner for the multi-view stereo Z-buffer as for the conventional single-view Z-buffer. However, naïve application of each stage of the Z-buffer rendering pipeline to each view in a multi-view stereo image independently of all other views would result in the work of many stages in the pipeline being replicated unnecessarily for each view. The algorithm described here attempts to make as much use as possible of stereo coherence between the views to eliminate redundant computations and reduce overheads wherever possible.

Based on the formulation of the autostereo perspective projection described in Section 4.3.3, it is possible to share much of the processing in the rendering pipeline for all views in a stereo image. How stereo coherence may be utilised at each stage of the rendering pipeline shown in Figure 5.1 is discussed in the sections which follow.

5.3 Traversal of the scene database

The aim of *traversing the scene database* is to obtain primitive object descriptions suitable for subsequent processing by the visible surface algorithm. The nature of this procedure is highly dependent on the representation of the scene database itself. Most databases may be described in terms of logical high-level entities (e.g. "car", "house", "chair") which represent complex arrangements of primitive structures

or even references to other high-level entities (e.g. "wheel", "roof", "leg"). To decompose this high-level representation into low-level rendering primitives may demand considerable effort, especially where there is a significant difference between the geometric complexity of the high- and low-level primitives. Even certain low-level entities may not be able to be represented directly in terms of primitives capable of being processed by the visible surface algorithm (e.g. curved surfaces such as spheres or cylinders in a system which only supports planar polygon primitives) and thus need to be approximated by suitable collections of primitives. On the other hand, more simple scenes may be represented by explicit instances of suitable primitives already and thus require little or no conversion at all.

The important feature of the database traversal process (as far as the Stereo Z-buffer is concerned) is that it is expected to produce the same set of rendering primitives regardless of any viewpoint-related factors — that is, it is a *view-independent* procedure. As the rendering primitives are the same in all views of a multi-view stereo image, there is no need for the database to be traversed separately for each view. The Stereo Z-buffer algorithm can capitalise on this by only traversing the database once regardless of the number of views in the image, effectively sharing the work done among all the views. In contrast, applying conventional single-view rendering techniques independently to each view in the stereo image would result in the traversal of the scene database being replicated redundantly for each view. As the number of views in the image increases, so the benefits of sharing the costs of such view-independent processing using the Stereo Z-buffer approach also increase. This is an example of perhaps the simplest form of stereo coherence, where improvements may be made "for free" by restructuring the rendering task specifically for stereo images.

5.4 Modelling transformations

Modelling transformations are generally used to specify the position, orientation, and scaling of an object in relation to other elements in the scene. How these transformations are applied is often closely linked in practice to the representation used for the scene database and how this is converted into rendering primitives ready for the visible surface algorithm, as outlined in Section 5.3. For example, hierarchical scene models typically have local modelling transformations applied at each level in the hierarchy, while very simple primitive-based scenes might not use any such transformations at all. The cost of this stage of the rendering pipeline depends largely on the geometric complexity of the database, which is directly related to the number of primitives and level of geometric detail represented.

The Stereo Z-buffer algorithm only needs to apply modelling transformations

once per object per stereo image, regardless of the number of views in the image. Depending on the geometric complexity of the scene and the number of views in the image, considerable savings may be made by eliminating the redundant processing of modelling transformations separately for each view. These savings are a natural by-product of restructuring the rendering algorithm for stereo, in a manner similar that described for database traversal in Section 5.3.

5.5 Illumination model

The *illumination model* describes how the interaction of light on the surface of an object produces a particular colour sensation in the eye of the observer. In the real world, this may depend on a great many factors, including the properties of the light sources, the properties of the surfaces, and the geometric relationships between them. These factors may interrelate in many ways to produce a variety of visual effects which the observer may interpret as dull matt or high-gloss reflection, brightly lit or in shadow, transparent or opaque, curved or flat, rough or smooth, and anything in between. As each primitive passes through the rendering pipeline, the illumination model is used to determine how it should appear in the image in terms of its colour, shading, and texture.

The challenge of reproducing the enormous visual wealth of the real world is an ongoing research topic in the field of synthetic image generation. In practice, any reasonably efficient implementation of a particular illumination model is often closely bound to the rendering algorithm as a whole. Pioneering work into illumination models was done in conjunction with some of the first hidden surface algorithms (see Gouraud [1971] and Phong [1975] for some of the best-known early models). A seminal paper by Whitted [1980] extended previous work to cover certain *global illumination* effects (such as reflection, refraction and shadowing) due to interactions between objects in the environment in a unified manner, similar to work by Kay and Greenberg [1979]. Further work on the modelling of global illumination using radiosity (Goral et al. [1984]; Nishita and Nakamae [1985]) provided a means of handling diffuse interactions missing from earlier approaches. An attempt to generalise these and other techniques and place them in a unified theoretical framework is presented by Kajiya [1986].

In practice, there is a trade-off between the quality of the image and the computational effort required to produce it. For the purposes of the renderer described here, only a simple illumination model will be considered.

5.5.1 A simple illumination model

While not purporting to be particularly accurate or realistic, a simple illumination model provides the opportunity to consider some of the fundamental characteristics of the problem. The model outlined in the following sections is based on a conventional model described in Foley et al. [1990], sections 16.1 and 16.2. It essentially considers only local illumination effects for each surface in isolation, ignoring the details of any interactions between surfaces on a global level.

In the discussions that follow, light intensity I is modelled as a unitless quantity without regard to its particular wavelength (and thus colour). This greatly simplifies the discussions and models monochromatic light adequately. In keeping with conventional colour models for computer graphics however, it should be noted that an acceptable approximation to coloured light may be obtained by sampling the colour spectrum at the wavelengths corresponding to the red, green, and blue primaries displayable on conventional CRTs. This is equivalent to applying each illumination calculation three times — once for each of the red, green, and blue samples — with the substitution of the appropriate wavelength-dependent parameters in each case.

5.5.1.1 Ambient light

Ambient light is something of a catch-all term which represents the global illumination contributions of all interacting objects in an environment. In this simple model, it is a grossly-oversimplified term which assumes that this global illumination can be treated as a diffuse, non-directional source of light which impinges equally on all surfaces in the environment with intensity I_a. The ambient light reflected from a particular surface is then

$$I = I_a k_a \tag{5.1}$$

where k_a is the coefficient of ambient reflection for that surface. In a more sophisticated illumination model such as radiosity, this term would be replaced by a much better approximation which actually attempts to model the highly complex set of diffuse interactions which produce it.

5.5.1.2 Diffuse reflection

Diffuse reflection refers to the reflection of light from a direct source onto a matt surface. Such a surface appears matt due to the fact that it reflects light equally in all

directions. The intensity of light reflected from a diffuse surface depends on the angle between the surface normal and the direction to the light source, as described by:

$$I = I_\ell k_d(\vec{N} \cdot \vec{L_\ell}) \tag{5.2}$$

where I_ℓ is the incident light intensity from a point source, k_d is the coefficient of diffuse reflection for the surface, $\vec{N}$ is the surface normal and $\vec{L_\ell}$ is the normalised direction from the point on the surface to the light source. Note that the scalar product term $\vec{N} \cdot \vec{L_\ell}$ in Equation 5.2 is equal to $\cos\theta$, where θ is the angle between the surface normal $\vec{N}$ and the direction to the light source $\vec{L_\ell}$. This simple model ignores any effects due to the attenuation of light intensity with distance from the source, although this could be incorporated in the I_ℓ term. It also does not explicitly address the possibility of self-shadowing, where the light source actually illuminates the side of the object opposite that which is facing the observer (that is, where $|\theta| \geqslant 90°$).

5.5.1.3 Specular highlights

A *specular highlight* may be observed when a light source illuminates a shiny surface. The position of the highlight depends on the position of the observer with respect to the surface and the light source, thus the highlight may appear to move when the observer moves. This effect is due to the fact that shiny surfaces reflect different amounts of light in different directions. For a perfectly shiny surface such as a mirror, light is reflected away from the surface at an angle equal but opposite to the angle of incidence. For less perfect reflectors however, light may be scattered somewhat around the ideal direction of reflection. A popular empirical approximation to this behaviour for non-perfect specular reflectors (Phong [1975]) is described by:

$$I = I_\ell k_s(\vec{R_\ell} \cdot \vec{V})^n \tag{5.3}$$

where I_ℓ is the incident light intensity from the point light source, k_s is the coefficient of specular reflection for the surface, $\vec{R_\ell}$ is the normalised direction of reflection of the light source at that point on the surface, $\vec{V}$ is the normalised direction of the viewpoint from the point on the surface, and n is the specular reflection exponent for the surface. Note that the scalar product term $\vec{R_\ell} \cdot \vec{V}$ in Equation 5.3 is equal to $\cos\alpha$, where α is the angle between the ideal reflection vector $\vec{R_\ell}$ and the direction to the viewpoint $\vec{V}$ at that point on the surface.

5.5.1.4 Combining illumination terms and multiple light sources

It is possible to combine the illumination contributions due to ambient, diffuse and specular reflection of light by a simple sum of the terms involved. Furthermore, this allows multiple point light sources to be accommodated by summing the diffuse and specular contributions from each light source. Combining the terms due to Equation 5.1, 5.2 and 5.3 from m point light sources produces the following:

$$I = I_a k_a + \sum_{\ell=1}^{m} I_\ell (k_d(\vec{N} \cdot \vec{L_\ell}) + k_s(\vec{R_\ell} \cdot \vec{V})^n) \tag{5.4}$$

5.5.2 Extending the illumination model to multiple views

In the simple illumination model outlined in the preceeding sections, only the specular highlight contribution depends on the position of the viewpoint. With this observation in mind, much of Equation 5.4 can be computed only once and re-used for all views, with only the horizontal component of the viewpoint direction $\vec{V}$ being linearly interpolated from one view to the next. Indeed, for essentially diffuse surfaces with little or no specular properties, even this term may be safely ignored and the illumination model evaluated once per point and re-used in all views in the image.

It is possible to separate the *view-dependent* components from the *view-independent* in other more sophisticated illumination models as well. For example, radiosity methods (Goral et al. [1984]; Cohen and Greenberg [1985]) generally compute a viewpoint-independent solution to the global illumination problem which may be re-used for all views, thus spreading the cost of the solution to some degree. Similarly, shadow effects due to occlusion of a surface from a light source do not depend on the position of the observer and thus the same shadow information may be used in all views. However it may be difficult to extract similar benefits with more view-dependent effects such as refraction (or even non-refractive transparency) and the aforementioned specular reflection.

5.6 Viewing transformation

The *viewing transformation* converts all points in the scene from world space coordinates to *viewing space* coordinates in the virtual camera frame of reference, given an arbitrary camera position and viewing direction. It is used to simplify the geometry of subsequent clipping, perspective projection and visible surface computations by presenting them with a single consistent view of the scene independent of the actual viewing geometry used.

With the autostereo perspective projection geometry described in Section 4.3.3, the viewing transformation only needs to be applied once per point in the scene per image generated, regardless of the number of views in the image. This eliminates the need to repeat the viewing transformation independently for each view, as all stereo cameras share a common frame of reference. Indeed, a multi-view stereo camera can be treated in essentially the same fashion as a conventional single-view camera for the purposes of the viewing transformation.

5.7 Clipping

Clipping is the process of determining which objects or parts of objects in the environment fall outside the bounds of the region of space represented by the image. Its purpose is to allow parts of the environment which cannot contribute to the image to be eliminated from further consideration in the later, more expensive stages of the rendering pipeline, particularly visible surface determination and rasterisation.

When clipping for a perspective projection of a three-dimensional environment, the aim is essentially to eliminate those parts of the environment which could not be seen directly from the viewpoint. There are two basic approaches to the problem:

viewing space clipping operates on objects whose geometry is expressed in three-dimensional viewing space coordinates, prior to the perspective projection. Objects are clipped against a region of space known as the *clipping volume*. This volume is typically constrained by clipping planes defined by the relationship of the viewpoint to the horizontal and vertical extents of the viewing window, as well as by an arbitrary depth extent.

image space clipping operates on projections of objects whose geometry is expressed in two-dimensional image space coordinates, after perspective has been applied. Object projections are clipped against the horizontal and vertical extents of the viewing window.

If viewing space clipping of objects is performed against a suitable finite viewing volume, subsequent image space clipping of object projections may not be necessary. This is typically the case when clipping for a single viewpoint. However, it may be more appropriate to perform some clipping operations in image space after perspective has been applied, particularly when clipping for multiple stereo views. These issues are examined in the following sections. For a general review of clipping and its relationship to the viewing model and projection geometry, see Foley et al. [1990], chapter 6.

5.7.1 Clipping for a single viewpoint

Clipping for a single viewpoint may be readily achieved by clipping against a suitable three-dimensional viewing volume. An example of such a volume in relation to a simple viewing model is shown in Figure 5.2. The viewing volume illustrated is finite, bounded by a truncated pyramid. The viewpoint is located where the apex of the pyramid would be found. The sloping sides of the pyramid pass through both the viewpoint and the horizontal and vertical extents of the viewing window. The base and roof of the truncated pyramid are formed by planes parallel to the viewing plane which bound the chosen depth extent of the viewing volume.

It is convenient to consider the clipping volume as the intersection of a set of unbounded clipping planes. The clipping planes formed by the viewpoint and the horizontal edges of the viewing window are known as the *top* and *bottom* clipping planes. Those formed by the viewpoint and the vertical edges of the viewing window are called the *left* and *right* clipping planes. Those formed by the the planes parallel to the viewing plane are referred to as the *near* and *far* clipping planes.

Using this model, objects may then be clipped against each bounding plane in turn until either all remaining parts of the object fall outside the clipping volume or all clipping planes have been examined. As an object is clipped against a given plane, the relationship of the object to the plane is considered. If the object lies entirely within the half-space of the plane inside the viewing volume, the object is retained unchanged. If it lies entirely outside this half-space, the object is considered invisible and is eliminated from further consideration. If it lies partially inside and partially outside the viewing volume half-space, the intersection of the object with the inside half-space is computed and this clipped object is used in place of the original object in subsequent processing. If any part of the object lies outside the clipping plane half-space containing the viewing volume, the object is said to be *clipped by the plane.*

The balance between clipping to a three-dimensional viewing volume in viewing space and clipping to a two-dimensional viewing window in image space is (to some degree) a matter of convenience. The absolute minimum three-dimensional clipping required is to a near clipping plane situated at (or a small distance in front of) the viewpoint, in order to eliminate any parts of the environment behind the viewpoint appearing in the image. Clipping to near or far depth planes can perhaps only be performed in viewing space, but the effect of clipping to the planes defined by the bounds of the viewing window can generally be achieved equally well in image space following the perspective projection. Figure 5.3 shows how an object clipped in viewing space to the left clipping plane projects to the same points as the projection of the object clipped in image space to the left of the viewing window. Other clipping planes follow the same pattern.

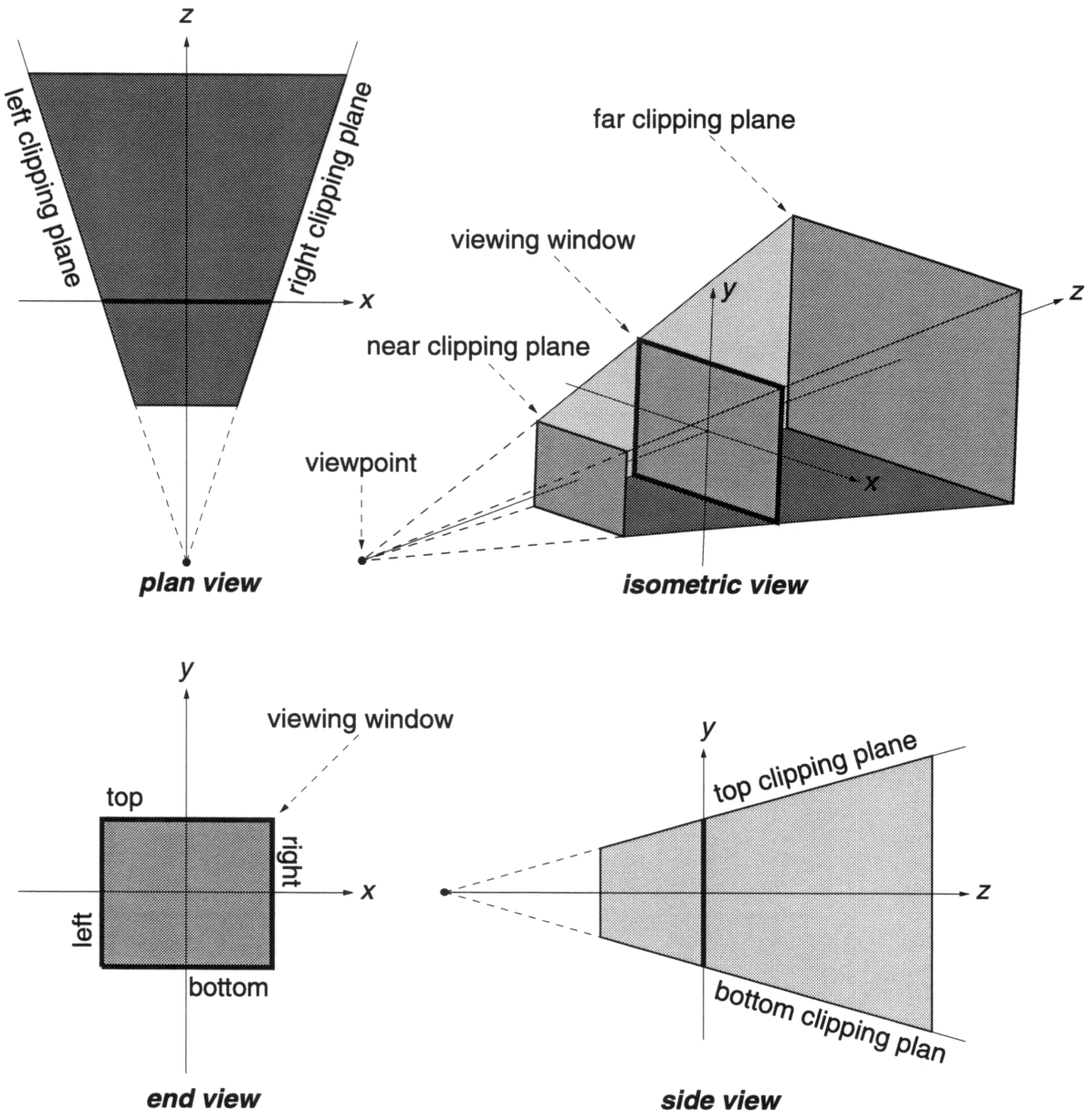


Figure 5.2: Clipping against a three-dimensional viewing volume for a single viewpoint

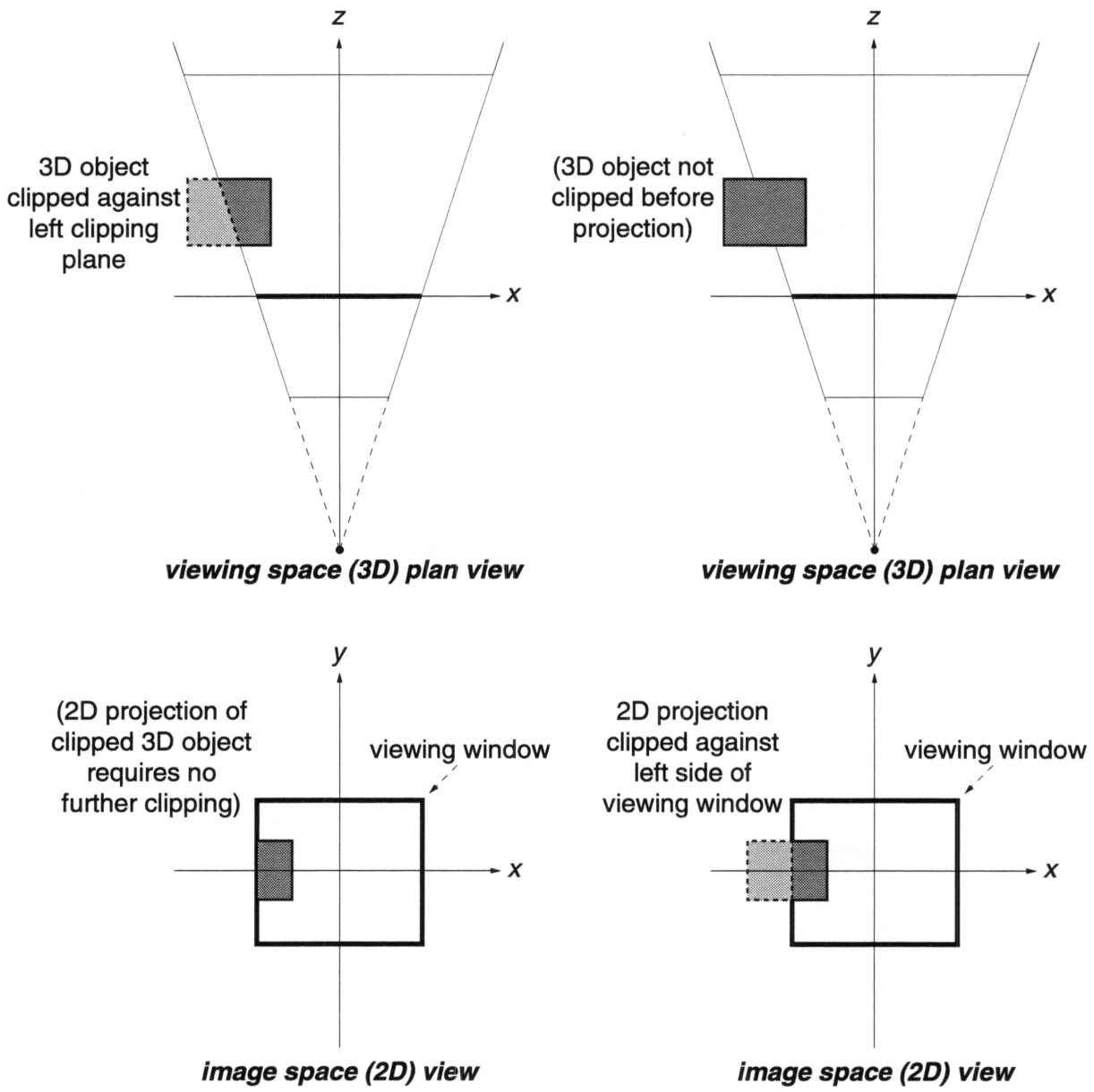


Figure 5.3: Comparison of viewing space and image space clipping. The 2D projection view seen by the observer inside the viewing window is the same in both cases.

5.7.2 Clipping for multiple stereo viewpoints

It is possible to extend the clipping techniques outlined in Section 5.7.1 directly for multiple viewpoints in stereo. Figure 5.4 shows an example of how this may be done for four stereo viewpoints; additional views may be handled in the same manner. It can be seen that the left and right clipping planes differ from one view to the next, owing to the horizontal displacement of the stereo viewpoints with respect to one another. As the views all share the same relative position of the top, bottom, near and far clipping planes, there are no such differences in these directions.

The differences in the viewing volumes for each viewpoint mean that the clipping of a given object in the environment may be different in each view. In particular, objects clipped by the left or right side of the clipping volume in one view may be clipped by a different amount (or even not clipped at all) in another view. On the other hand, objects clipped by the top, bottom, near or far clipping planes will be clipped the same with respect to these planes in all views. This suggests that an efficient method of clipping for multiple stereo views should share the results of clipping to the top, bottom, near and far clipping planes common to all views, only performing clipping to the left and right clipping planes separately for each view.

It is important to consider how this clipping might be performed in practice and what (if any) effect it may have on other stages of the stereo rendering pipeline. If an object is clipped against the left and right clipping planes for each view independently before perspective is applied, the object may be clipped differently in each view. With different clipped versions of the object, it may be difficult (if not impossible) to exploit stereo coherence between the perspective projections of the object in each view as described in Section 5.8. If such clipping takes place after perspective projection but before scan conversion, it may be similarly difficult to take advantage of stereo coherence in the manner described in Section 5.9 when actually scan converting the object. Conversely, it may prove unacceptably inefficient to perform all clipping to the left and right sides of the viewing window only after perspective projection, owing to the large number of objects which may be situated so far off to the sides that they do not appear in any view.

Therefore a compromise is proposed whereby clipping to the sides of the viewing region is achieved in two successive stages. The first stage performs approximate clipping of objects in viewing space to left and right clipping planes that are arranged to cover a volume wide enough to encompass the left and right clipping planes of all the views. See Figure 5.5. The purpose of this step is to perform a gross culling operation to eliminate from further consideration those objects which do not appear in any view. Thus strict accuracy is not required, provided conservative estimates of visibility are used. The second stage clips projected objects more precisely to the left and right bounds of the viewing window. This may be performed

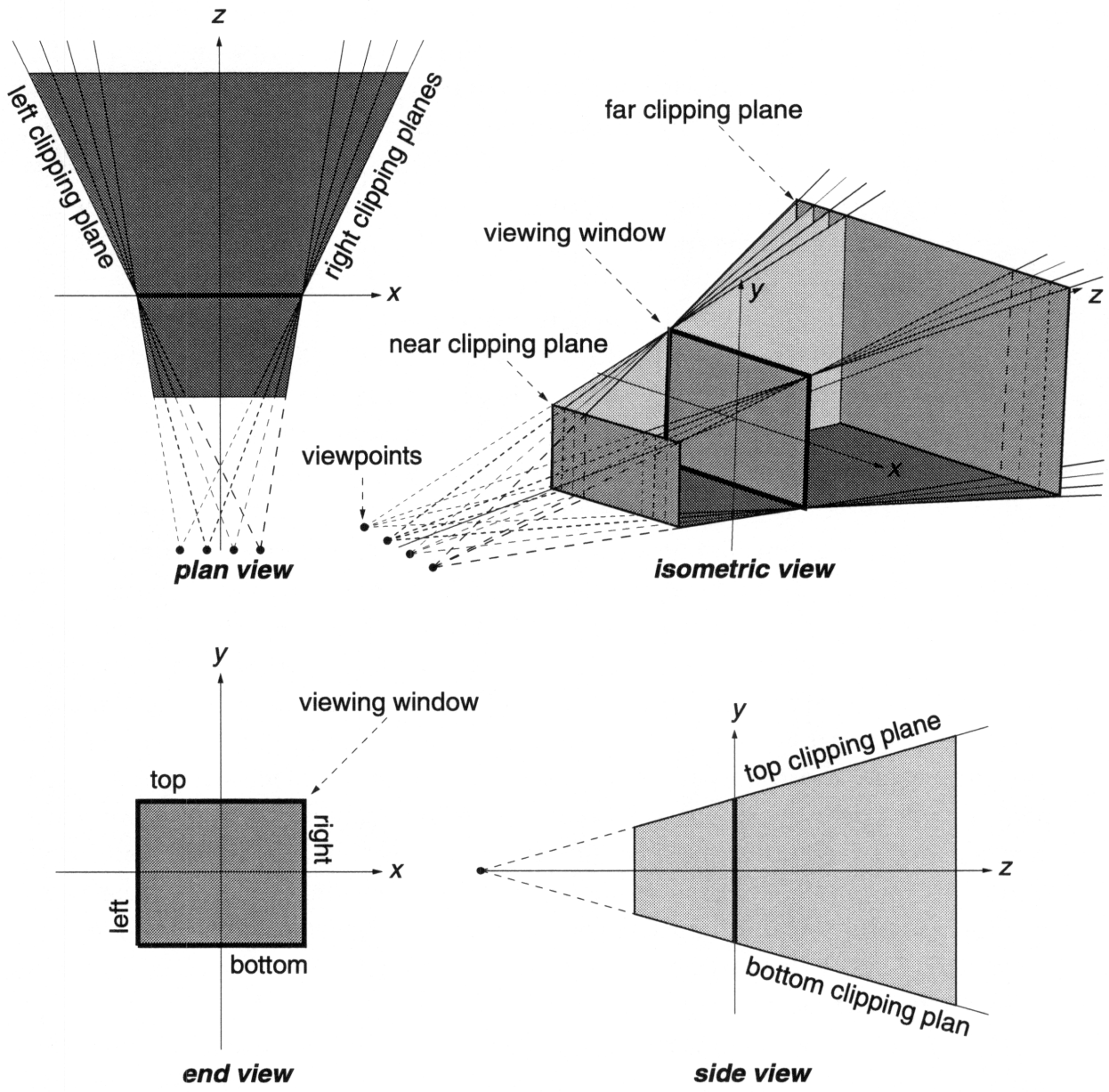


Figure 5.4: Clipping regions for multiple stereo views. The left and right clipping planes differ between the views, while the top, bottom, near and far planes are common to all views.

either as part of the scan conversion process (using image space clipping) or as a pre-processing step immediately prior to scan conversion (using image space or viewing space clipping), as appropriate to the rendering technique used.

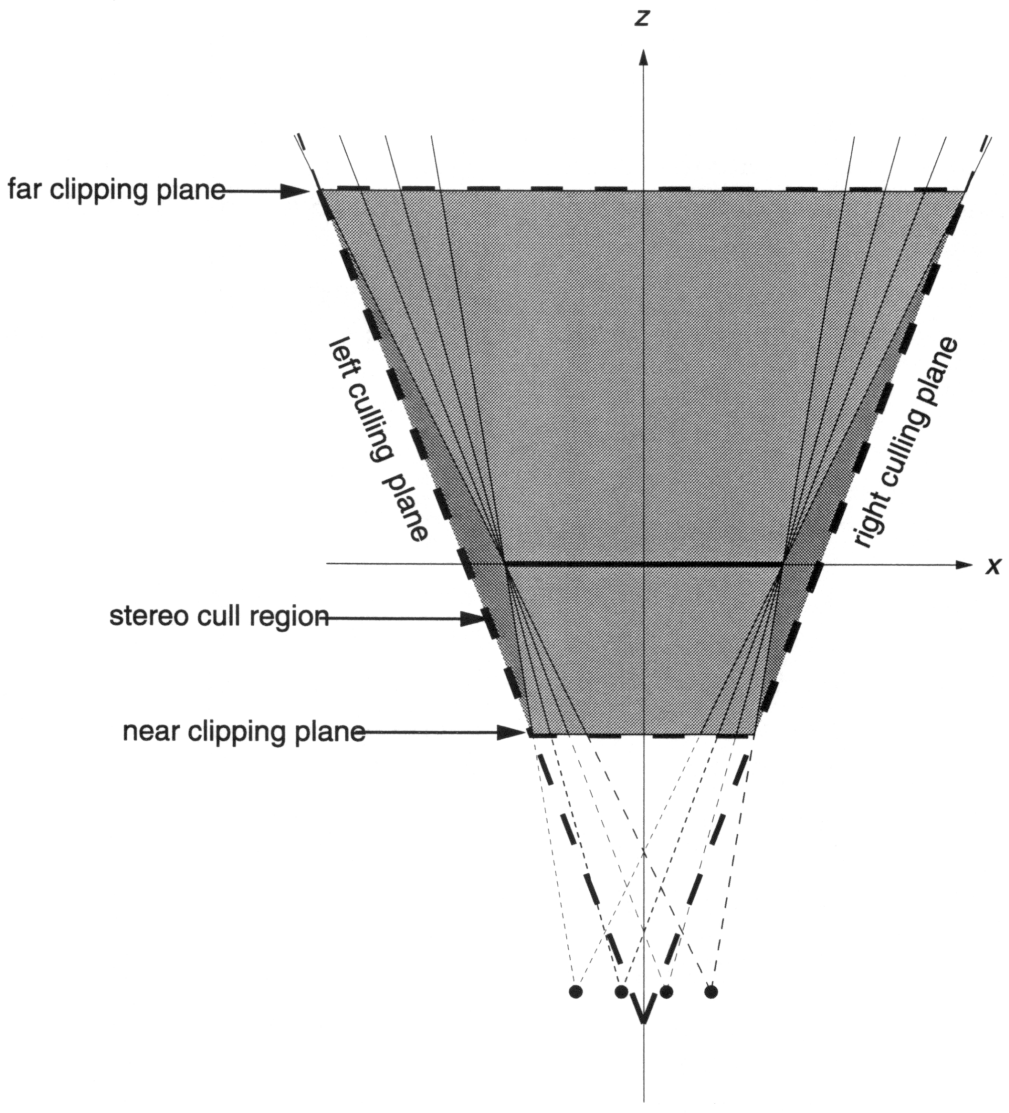


Figure 5.5: Approximate clipping to left and right stereo boundaries

5.8 Perspective projection

Perspective projection is used to map each three-dimensional object onto the image plane, as seen from a given viewpoint. This then allows the projected object to be treated in a similar manner to conventional two-dimensional rendering primitives in the subsequent rasterisation stage of the pipeline. In a stereo image, coherence between the views provides considerable scope for improving the efficiency of the perspective projection when applied to multiple views.

Consider the autostereo projection geometry described in Section 4.3.3. The expression for the projected vertical coordinate given in Equation 4.10 is the same for all views and thus we may benefit from computing its value only once. The expression for the projected horizontal coordinate given in Equation 4.9 varies from one view to another only with the viewpoint position, thus presenting further opportunities for savings due to coherence between the views. Let the horizontal parallax of a point visible in two views i and j be defined as the difference between the horizontal components of the projection of that point in each view, as in Equation 5.5:

$$\begin{aligned} h_{i,j} &= x_i - x_j \\ &= \frac{Z \cdot (e_i - e_j)}{d + Z} \end{aligned} \tag{5.5}$$

Rearranging Equation 5.5 enables us to compute the horizontal projection of a point in view i as $x_i = x_j + h_{i,j}$, given an initial projection in view j and the offset from viewpoint j to i. If $j = i-1$, then an incremental computation of the horizontal projection in each successive view is possible, given an initial projection and the offset from each viewpoint to its successor. Furthermore, if the viewpoints are all equally spaced, then $\Delta e = e_i - e_{i-1}$ is a constant and thus $h_{i,i-1}$ is also a constant h for a given point P, which allows the horizontal projection of that point in each view to be computed by linear interpolation across the views. This is the basis of the approach described below.

Consider a set of n viewpoints $C_1, C_2, \ldots, C_n$ which are arranged as described in Section 4.3.3. The total separation between the leftmost and rightmost viewpoints is Δe and viewpoint i is located at $C_i(e_i, 0, -d)$, where $e_i = (i - \frac{n+1}{2}) \cdot \Delta e$. For the first view, Equations 4.9 and 4.10 are applied with $i = 1$, to compute the horizontal and vertical components (x_1 and y_1 respectively) of the projection of a point $P(X, Y, Z)$ in viewing space onto the viewing plane. See Equations 5.6 and 5.7. For each successive view i (where $2 \leqslant i \leqslant n$), the horizontal coordinate of the projected point (x_i) is obtained by adding the horizontal view-to-view parallax (h) of that point to the horizontal coordinate in the previous view (x_{i-1}). The vertical coordinate (y_i) is the same as in the first view (y_1). See Equations 5.8, 5.9 and 5.10.

$$x_1 = \frac{X \cdot d - Z \cdot \frac{n-1}{2} \cdot \Delta e}{d + Z} \tag{5.6}$$

$$y_1 = \frac{Y \cdot d}{d + Z} \tag{5.7}$$

$$h = \frac{Z \cdot \Delta e}{d + Z} \tag{5.8}$$

$$x_i = x_{i-1} + h \tag{5.9}$$

$$y_i = y_1 \tag{5.10}$$

Without performing any sharing of common subexpressions, it is possible to calculate the coordinates of the projection of the point in the first view in 11 floating point operations (FLOPs). To compute the coordinates of the projected point in subsequent views, there is an initial cost of 3 FLOPs to obtain the view parallax increment, but then each additional view costs only a single FLOP to perform the parallax interpolation. Thus an unoptimised n-view autostereo perspective projection utilising coherence can be performed at a total cost of $13 + n$ FLOPs. Optimising the computation by sharing common subexpressions could reduce this cost to $8+n$ FLOPs. On an average cost-per-view basis this compares well with a similarly optimised conventional single-view perspective projection at a cost of 4 FLOPs, with the optimised autostereo projection costing less ($1 + 8/n$ FLOPs per view) for 3 or more views. As the number of views increases, so the cost-per-view goes down asymptotically towards 1 FLOP: for 8, 16 and 32 views the optimised autostereo projection cost is 2, 1.5 and 1.25 FLOPs per view respectively. In certain circumstances however, it may even be possible to use the same approach to eliminate altogether the explicit calculation of the perspective projection of each vertex of a primitive in every view, as will be shown in Section 5.9.

5.9 Scan conversion

Scan conversion is the process of mapping low-level graphical primitives onto the individual pixels which represent them in the image. Polygons are perhaps the most common low-level graphical primitive in use with Z-buffer rendering systems. A polygon may be described by an ordered list of its vertices, with edges from one vertex to the next implicitly defined by the ordering. In the discussion that follows, polygons are assumed to be planar and non-self-intersecting; that is, all points in a particular polygon lie in the same plane, and no edge of the polygon crosses another.

The image is assumed to be stored in a random access frame buffer memory, where the location of each pixel in the image corresponds to a particular address in the frame buffer. Generally, an image is made up of a number of horizontal lines of pixels stored contiguously in memory, corresponding to the way in which the image is read out of the frame buffer for display on a typical raster scan CRT device.

For a stereo image, it may be possible to take advantage of the similarities between the projections of an object in each view (as outlined in Section 5.8) to simplify the scan conversion process. How stereo coherence may be applied to polygon scan conversion in a multi-view stereo image is described in this section.

Consider first how a polygon may be scan converted for a single view. The viewing space coordinates of the polygon vertices are projected onto the viewing plane according to the perspective calculation described by Equations 4.9 and 4.10. This produces a two-dimensional projection of the original three-dimensional polygon, as seen from the appropriate viewpoint. The polygon may now be scan converted as a two-dimensional primitive. One algorithm for scan converting a polygon is outlined in Figure 5.6, which is similar to a method described in Foley et al. [1990], section 3.6.

```
        procedure scanpoly(vertex[num_vertices]) {
(1)     .   sort edges from top to bottom
(2)     .   initialise active edge table to be empty
(3)     .   for each horizontal scanline between top and bottom of polygon {
(4)     .   .   check for any edges which start or finish on current scanline
                  and update active edge table accordingly
(5)     .   .   find intersections of active edges with current scanline
                  and match left/right edges to form interior horizontal spans
(6)     .   .   for each span in current scanline {
(7)     .   .   .   for each pixel in span {
(8)     .   .   .   .   if pixel is visible {
(9)     .   .   .   .   .   shade pixel
                        }
                    } next pixel
                } next span
            } next scanline
        } end scanpoly
```

Figure 5.6: Pseudocode for polygon scan conversion algorithm. Steps are numbered in the leftmost column for reference.

The algorithm in Figure 5.6 converts a polygon represented by a list of its vertices into a discrete set of pixels which lie inside the boundary of the polygon. By

incrementally updating information about the intersection of each edge in the polygon with each horizontal pixel scanline in the image, it exploits image space coherence to simplify the task of determining which pixels are inside the polygon. The set of intersections of polygon edges which are active (i.e. present) on a given scanline allows contiguous spans of interior pixels to be identified and shaded, subject to the polygon's visibility at each pixel.

After sorting the edges vertically in Step (1) and initialising the active edge table in Step (2), the polygon is processed one horizontal scanline at a time in the loop of Step (3). This allows both the active edge table and the edge intersections to be updated in an incremental manner in Steps (4) and (5) respectively. The interior pixels of each horizontal span obtained from Step (5) are scanned in the loops of Steps (6) and (7). Each pixel of each span is conditionally shaded in Step (9), based on the result of the visible surface test in Step (8).

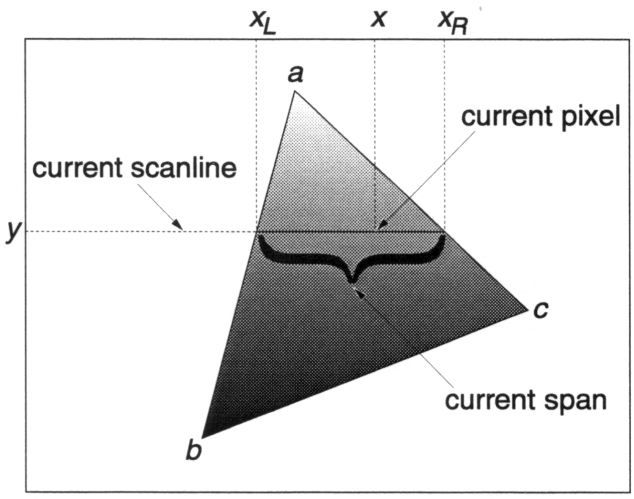


Figure 5.7: Scan conversion of a simple polygon. Pixel x lies on the span between x_L and x_R, along the intersection of scanline y with active edges ab and ac.

An example of how the algorithm in Figure 5.6 is used is illustrated in Figure 5.7. The polygon shown has vertices a, b, and c. The edges between the vertices (sorted from top to bottom) are ab, ac, and cb. Edges ab and ac are active in scanline y; edge cb has not yet been reached in the top to bottom scan order. The intersection of edge ab with scanline y is x_L, and the intersection of edge ac with scanline y is x_R. Together these active edge intersections form the only span on scanline y, from x_L to x_R. Individual pixels in the span may then be processed by conventional Z-buffer visible surface determination techniques, and shading applied to any visible pixels.

Now consider how stereo coherence may be taken advantage of to speed up scan conversion of polygons in more than one view to produce a stereo image. Using the results derived in Section 5.8, it can be seen that the vertical component of the perspective projection of any point as seen in stereo is the same in all views. This can readily be incorporated into the polygon scan conversion algorithm outlined in Figure 5.6 by sharing any processing related to the vertical components of the polygon's geometry between the views.

One algorithm which uses this method to speed up scan conversion of a polygon in two stereo views is described by Adelson et al. [1991]. The basic approach to polygon scan conversion used by this algorithm is very similar to that outlined in Figure 5.6, but the vertical sorting of vertices and updating of the active edge table is shared by both views, thus saving on the overheads of Steps (1) to (4). However, active edge intersections with the current scanline are stored and updated independently for each view (corresponding to Step (5)), without any savings being made. Furthermore, the scanning of pixels over the horizontal extent of each span (Steps (6) to (9)) must be done separately for each view, again producing no savings. This is due to the fact that a span formed between two given edges may have different widths in each view, depending on the difference in depths (and thus stereo parallax) at the intersection of the edges with the current pixel line.

Naïvely adapting this algorithm to polygon scan conversion for multi-view stereo is straightforward, but may not necessarily lead to the most efficient implementation. One potential problem concerns the way in which information about the intersection of an edge with the current scanline is maintained and updated independently for each view. While this may be acceptable for the original two-view case, as the number of views increases so do the overheads of such an approach, particularly in terms of the storage required to maintain edge intersection and slope information for the image of each edge in each view. If a renderer capable of dealing with a variable number of views is required — as is the case with the current experimental system which uses anything from one to sixteen views — the need for an alternative solution is even more pressing.

The method proposed here is based on the idea of extending the use of stereo parallax to interpolate the perspective projection of not only the polygon vertices as described in Section 5.8, but also the intersection of the polygon edges with each scanline. How this may be achieved using only linear interpolation and with a fixed amount of storage regardless of the number of views will now be described.

Consider the edge shown in Figure 5.8. Applying the perspective projection geometry described in Section 4.3.3, the edge endpoints $P_a(X_a, Y_a, Z_a)$ and $P_b(X_b, Y_b, Z_b)$ project to $Q_{a_i}(x_{a_i}, y_{a_i}, 0)$ and $Q_{b_i}(x_{b_i}, y_{b_i}, 0)$ in view i. As the vertical coordinates of a projected point are the same in all views of a stereo image, it is possible to write y_a for y_{a_i} and y_b for y_{b_i}.

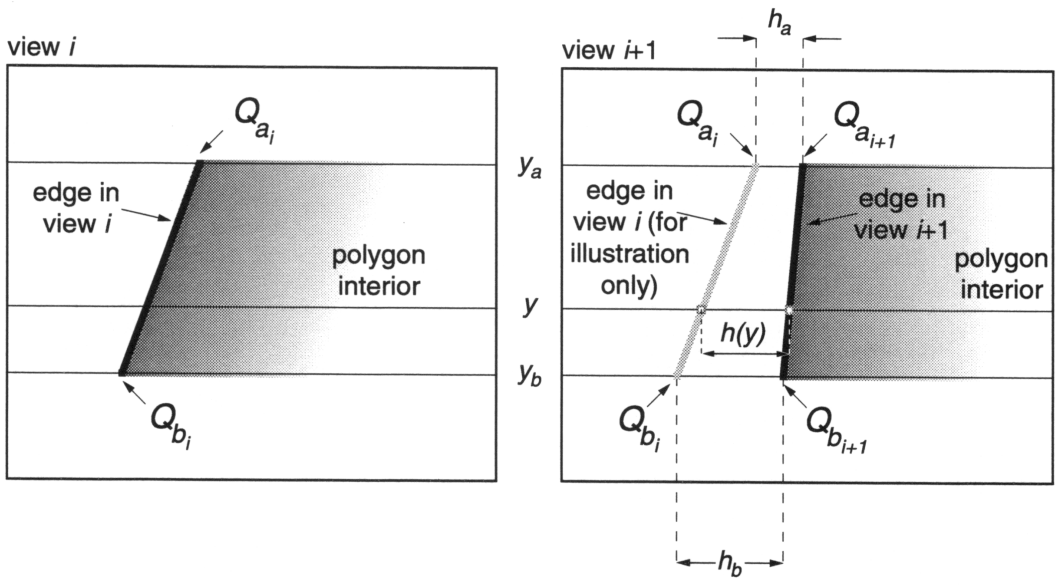


Figure 5.8: Interpolation of the intersection of an edge with a horizontal scanline in adjacent stereo views

Using Equations 4.9 and 4.10, the coordinates of the projected edge endpoints Q_{a_i} and Q_{b_i} in view i are given by:

$$\begin{aligned} x_{a_i} &= \frac{X_a \cdot d + Z_a \cdot e_i}{d + Z_a} \\ y_a &= \frac{Y_a \cdot d}{d + Z_a} \\ x_{b_i} &= \frac{X_b \cdot d + Z_b \cdot e_i}{d + Z_b} \\ y_b &= \frac{Y_b \cdot d}{d + Z_b} \end{aligned}$$

If it is assumed that the viewpoints are equally spaced a distance Δe apart as described in Section 5.8, the horizontal stereo parallax of the endpoints between any views i and $i + 1$ is a constant for a given point (see Equation 5.8). Thus the horizontal stereo parallax of the endpoints of the line between views i and $i + 1$ are given by Equations 5.11 and 5.12:

$$\begin{aligned} h_a &= x_{a_{i+1}} - x_{a_i} \\ &= \frac{Z_a \cdot \Delta e}{d + Z_a} \end{aligned} \tag{5.11}$$

$$\begin{aligned} h_b &= x_{b_{i+1}} - x_{b_i} \\ &= \frac{Z_b \cdot \Delta e}{d + Z_b} \end{aligned} \tag{5.12}$$

An equation which gives the intersection of the edge in view i with a horizontal scanline y as a function of y is:

$$x_i(y) = x_{a_i} + (y - y_a) \cdot m_i \tag{5.13}$$

where $m_i = (x_{b_i} - x_{a_i})/(y_b - y_a)$ is the reciprocal slope of the edge[1] in view i. A similar equation may be written for the intersection of the edge in view $i + 1$ with the same horizontal scanline y. Subtracting one from the other gives the following expression for the stereo parallax between the intersection of the edge in views i and $i + 1$ with scanline y:

$$h(y) = x_{i+1}(y) - x_i(y) \tag{5.14}$$

Substituting the expressions for the edge intersections from Equation 5.13 into Equation 5.14 gives:

$$\begin{aligned} h(y) &= x_{a_{i+1}} - x_{a_i} + (y - y_a) \cdot (m_{i+1} - m_i) \\ &= h_a + (y - y_a) \cdot \frac{(x_{b_{i+1}} - x_{a_{i+1}}) - (x_{b_i} - x_{a_i})}{y_b - y_a} \\ &= h_a + \frac{y - y_a}{y_b - y_a} \cdot (h_b - h_a) \end{aligned} \tag{5.15}$$

using the definitions of h_a and h_b from Equations 5.11 and 5.12. Rearranging Equation 5.14 for $x_{i+1}(y)$ provides a way of obtaining the intersection of an edge with a horizontal scanline (y) in one view, given the intersection in the previous view ($x_i(y)$) and the stereo parallax of the edge at the intersection ($h(y)$). Thus it is possible to obtain the edge intersection point in one view by evaluating Equation 5.13 once only, and linearly interpolate the intersections in the remaining views using the interpolated stereo parallax from Equation 5.15.

With this formulation, the cost of calculating the intersection between an edge with a given horizontal scanline in n stereo views is one edge intersection interpolation (using reciprocal slope information for the edge) in one view, plus one stereo

[1] it is assumed that the vertical coordinates of the projected endpoints of the edge are different, and thus $y_b - y_a \neq 0$; that is, the perspective projection of the edge is not precisely horizontal. As horizontal edges may be safely ignored in the polygon scan conversion algorithm described in Figure 5.6, this is no restriction.

parallax interpolation (along the edge) for that scanline, followed by $n-1$ incremental edge intersection computations (using the interpolated stereo parallax value) for the remaining views. This actually costs one extra interpolation per scanline than the original independent edge intersection calculation, which only required n interpolation to compute edge intersections independently for each view. However, this cost is offset to some extent by the fact that only two interpolant initialisations are required per edge (one for the edge intersection point and one for the stereo parallax value) regardless of the number of views, instead of n interpolant initialisations (one per edge per view). Indeed, as initialising each interpolant generally requires several floating point operations (including a division), the parallax interpolation approach may even be more computationally efficient, especially where the number of views is relatively large and the number of scanlines in each edge of a polygon is relatively small. In addition, the proposed method also reduces the storage required to maintain the interpolated values, from the original n interpolant records down — one per view — to a constant two, regardless of the number of views.

The advantages of the parallax interpolation approach depend on the implementation context. For example, it may be possible to produce a lower-cost hardware design using the proposed parallax interpolation method, without imposing an arbitrary restriction on the number of views which may be generated. It also may prove beneficial for a software implementation where efficient memory usage is important, such as in a scan line hidden surface algorithm. In any case, the method is able to utilise stereo coherence to share the set-up costs of the vertical ordering of vertices and the overheads associated with maintaining the active edge table during the scan conversion process.

5.10 Visible surface determination

Visible surface determination has historically been considered the single most expensive stage in the entire rendering pipeline, as witnessed by the many and varied attempts to solve the problem (Sutherland et al. [1974]). Even with more recent developments in the field, the question of visibility between two points in an environment remains central to many issues in synthetic image generation (Kajiya [1986]).

Although visible surface determination at the pixel level in the Stereo Z-buffer is achieved by conventional techniques, it is possible to spread the cost of some of the overheads associated with Z-buffer calculations for polygon primitives among the views by taking advantage of stereo coherence. With the off-axis projection geometry used in Section 5.8, any given vertex in a polygon is the same distance from the viewing plane in all views in the stereo image, and thus the same Z-value may be used for that vertex in all views. Furthermore, the same principle allows Z-values

interpolated along an edge from one horizontal scanline to the next to be used for that edge in all views. At the span level however, the differences in the positions of the span endpoints between the views make it difficult to make further use of stereo coherence during processing of individual pixels in the span itself.

Assuming the stereo polygon scan conversion technique described in Section 5.9 is being used, the cost of the Z-value calculations may be readily shared in this way at all levels except the actual pixel-by-pixel visible surface step itself. The relative benefits of such sharing depend on two main factors: the number of views the edge information is shared between, and the ratio of the number of edge pixels to interior pixels in a polygon. The more views in the image, the greater the savings obtained by sharing the interpolated edge information among all views. The smaller the ratio of edge pixels to interior pixels, the more that the computations needed for interior pixels tend to dominate the overall Z-value interpolation cost and thus the smaller the advantages. Conversely, for smaller polygons the greater the relative cost attributable to edge pixels and the better the stereo method performs in comparison. This is a similar effect to that reported by Adelson et al. [1991] concerning the sharing of interpolated Gouraud shading values along a polygon edge between two stereo views. However the maximum potential benefits are considerably greater in this case, as the number of views possible with the proposed system is not limited to two.

5.11 Shading and texture mapping

Shading and texture mapping are two techniques which may be used to enhance the visual appeal and realism of synthetic images at a relatively low cost. Shading varies the intensity used to represent a graphical primitive in the image within the primitive itself, in such a way as to simulate the effects of changes of curvature on the illumination of the primitive. Texture mapping is used to map an arbitrary external image (referred to as the *texture map*) onto the surface of a graphical primitive, producing an effect similar to that of "wallpapering" the texture map image onto the primitive. Both techniques add visual complexity to the surface of the primitive, without increasing its geometric complexity. However, both add considerable cost at the rasterisation level as the associated shading or texture parameters must be calculated for each pixel. The following sections describe how these calculations may be performed in general, and how stereo coherence can be utilised to improve the overall efficiency of the shading and texture mapping process.

5.11.1 Interpolated shading

Interpolated shading is a relatively simple way to obtain smoothly shaded primitives. It simulates the effects of illuminating the primitive as if it had a smoothly curved (as opposed to geometrically flat) surface, including diffuse reflection and specular highlights. It works by interpolating the shading parameters between the vertices of the primitive to cover the interior points of the primitive. See Figure 5.9.

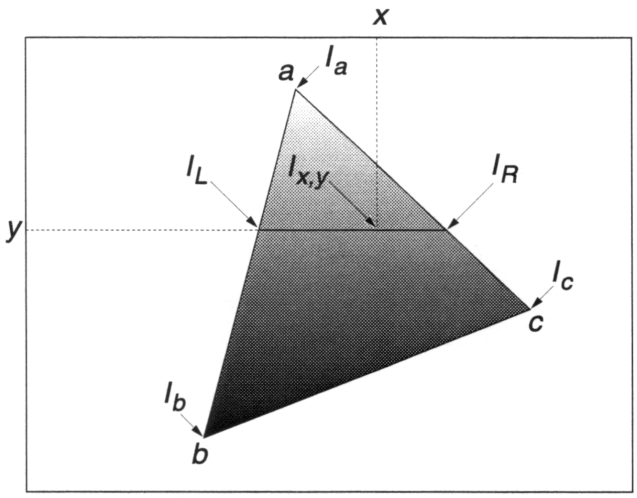


Figure 5.9: Simple interpolated shading. Intensity I_L is interpolated between I_a and I_b along edge ab, I_R is interpolated between I_a and I_c along edge ac and between I_c and I_b along edge cb. Intensity $I_{x,y}$ is interpolated between I_L and I_R along each scanline y between y_a and y_b.

Unfortunately, most interpolated shading models suffer from a number of undesirable artifacts, the nature of which may vary depending on which shading parameters are used in the interpolation. Gouraud shading (Gouraud [1971]) linearly interpolates intensity values between the vertices and across the interior of each polygon primitive, where the vertex intensities are obtained by evaluating the illumination model at each vertex. The appearance of a large complex curved surface may be approximated using a tessellated array of polygons, where shared vertices use common vertex normals for the evaluation of the illumination model in order to maintain continuity of shading values, although the apparent shading gradient may not be continuous between primitives. While it can produce acceptable results for small primitives or mostly diffuse surfaces, it suffers from a susceptibility to Mach band effects (Ratliff [1965]) along primitive boundaries owing to discontinuities in intensity gradients between primitives, as well as being unable to detect specular highlights in the interior of large primitives.

Phong shading (Phong [1975]) attempts to address this problem by interpolating the vertex normals instead, and by applying the illumination model at each individual pixel in the primitive. Although it is capable of producing much better results than simple Gouraud shading, it too is only an approximation and discontinuities in shading gradient may still be observed in certain circumstances. In addition, the need to renormalise the interpolated surface normal vector and apply the illumination model at each pixel makes it a significantly more expensive approach than Gouraud shading.

Another artifact common to interpolated shading techniques which has particular relevance to stereo images is the distortion of perspective which results from linear interpolation of the shading parameters in screen space rather than object space (Foley et al. [1990], section 16.2.6). Such distortion can produce changes in the apparent curvature of the object's surface, which may conflict with that intended to be suggested by the interpolated shading in the first place. In practice the effects of this distortion may be quite subtle and difficult to detect in a single (monoscopic) image, although the problem is often readily observed in an animated sequence (or in a stereo image). Perspective interpolation of shading parameters can be used to alleviate this problem (albeit at considerable cost), although other related problems remain such as the dependency of the interpolation on the orientation of the primitive. Despite these drawbacks, simple linear interpolation of shading parameters is often considered an acceptable trade-off between cost and quality.

Regardless of which shading parameters are used, there is some scope for sharing interpolated shading information between the views in a stereo image. In the same way in which Z-value calculations may be shared among all views at the polygon and edge level as outlined in Section 5.10, so can any view-independent component of the shading calculations also be shared. This means that for an illumination model such as that described in Section 5.5.1, the interpolation of any of the non-specular components along the edges of a polygon may be shared between all the views. View-dependent specular shading may strictly require re-evaluation in each view, although given the level of approximation inherent in most interpolated shading techniques there is an argument for ignoring the differences in specular components between the views for the sake of simplicity. Such an approximation would preclude the possibility of portraying sophisticated binocular illumination effects such as glitter, sparkle, or lustre (Lipton [1982], page 189), but these may be safely ignored with simple illumination models such as that described in Section 5.5.

5.11.2 Texture mapping

Texture mapping is a simple method for adding apparent detail to the surface of graphical primitives. It works by mapping an arbitrary image onto the primitive, where the detail in the image is used to modulate the appearance of the primitive at each point on its surface. A simple approach to texture mapping for planar polygons is illustrated in Figure 5.10.

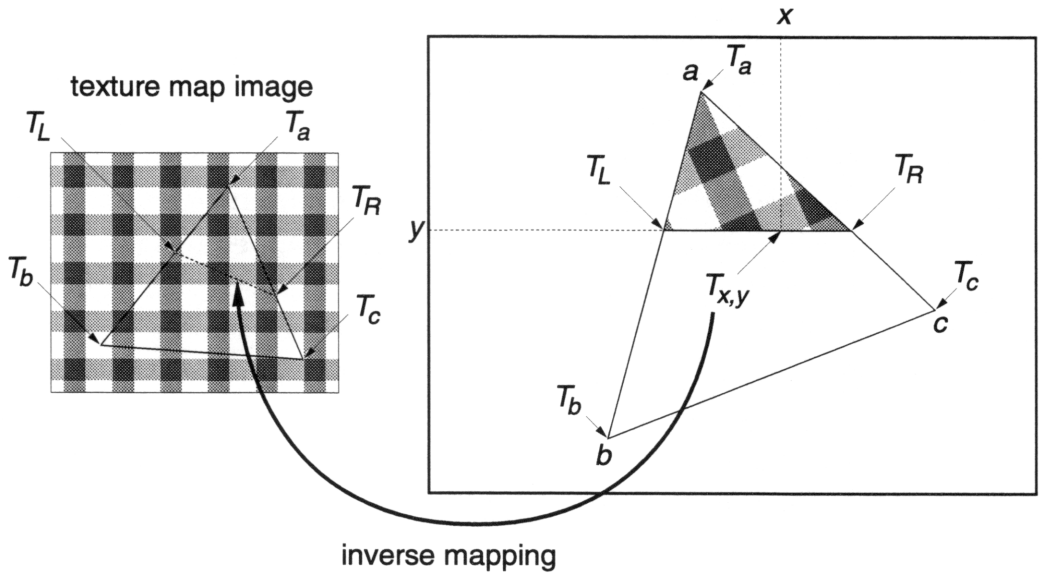


Figure 5.10: A simple model of texture mapping. Texture vertices are found by mapping polygon vertices onto texture map image. During scan conversion of the polygon, texture reference T_L is interpolated between T_a and T_b along edge ab, T_R is interpolated between T_a and T_c along edge ac and between T_c and T_b along edge cb. Texture reference $T_{x,y}$ is interpolated between T_L and T_R along each scanline y between y_a and y_b. For point sampling, the actual texel to be used at pixel (x, y) is found by inverse mapping $T_{x,y}$ back to the original texture map image.

First, an image to be used for the texture map is selected. Next, the vertices of the polygon are projected onto the texture map as if the texture map was coplanar with the polygon. The texture space coordinates of the projections of the polygon vertices onto the texture map are referred to as the *texture vertices*. During scan conversion, these texture coordinates are interpolated along the edges of the polygon between vertices and along spans between span endpoints, in much the same way as Z-values or shading parameters. The interpolated texture coordinate for a given pixel in the polygon provides an inverse mapping back to a point in the original texture map image which can be used to determine the intensity of the pixel.

In a simple point-sampling system, the texture reference would be used to select a single pixel (called a *texel*) from the texture map which is used directly as the intensity of the pixel in the polygon. However such a naïve approach ignores the problems of aliasing and blockiness which almost invariably results owing to mismatches in the size and shape of pixels in the perspective projected polygon and texels in the texture map. A variety of different filtering techniques have been reported in the literature to address this problem (Williams [1983]; Crow [1984]). While it is not appropriate to discuss these issues here, a good review of texture mapping in general which includes comparisons of a number of these techniques may be found in Heckbert [1986].

As with shading parameters, care must be taken with the interpolation of texture coordinates during scan conversion if unwanted distortion effects are to be avoided. Unlike the case for shading parameters however, perspective interpolation is essential for texture coordinates, as the distortions introduced by linear interpolation are generally much more noticeable in texture mapped images than with smoothly shaded surfaces. The additional computational cost of perspective interpolation may be offset to a certain extent by formulating the interpolation so that the same interpolation parameter may be used to compute more than one shading or texture parameter at each interpolation step, as suggested by Heckbert and Moreton [1991]. This is the approach taken in the current implementation, where the same perspective interpolation parameter may be used for both shading calculations and texture coordinates.

Texture coordinate calculations for stereo primitives can use stereo coherence between the views to share information among all views in the same manner as Z-values, as described in Section 5.10. Thus the interpolated texture coordinate references for each active edge only need to be calculated once per scanline, regardless of the number of views in the image. This is especially important given the relatively high computational cost of the perspective interpolation required for the texture references, although it is not as significant with relatively large, wide polygons where the calculations required for interior pixels tends to overshadow any savings made along the edges.

5.12 Multi-pass antialiasing

Although not strictly part of the main rendering pipeline itself, *multi-pass antialiasing* is a technique which combines the results of a number of separate point-sampled rendering passes to improve the quality of an image. A point-sampling renderer typically samples the scene database at the centre of each pixel. For multi-pass antialiasing, the location of the sample point within the pixel is varied for each pass.

The point-sampled image thus obtained is then suitably filtered and integrated into the composite image accumulated from all previous passes.

Varying the sample point on a regular subpixel grid implements a form of multi-pass supersampling. Distributing sample locations according to a non-uniform sampling strategy (see Section 3.3.2.4) allows aliasing artifacts to be readily traded for less objectionable noise in the image (Cook [1986]). If the accumulated image is displayed after each rendering and integration pass, an effect similar to progressive refinement is achieved, with the image quality improving over time as each successive sample image is incorporated into the composite result.

The principal advantages of such an approach to antialiasing are its simplicity and flexibility. It simplicity is due to the fact that it may be achieved by a straightforward combination of simple component subprocesses, namely point sampling and postfiltering. Its flexibility has been demonstrated by its ability to be used not only for spatial antialiasing but also to be applied to a variety of sophisticated rendering effects, including motion blur, depth of field, soft shadows, and translucency (Haeberli and Akeley [1990]; Mammen [1989]). This follows on from the work described by Cook [1986] and allows the multi-pass/multi-sample approach to be applied to image synthesis problems which would otherwise prove difficult to solve using analytic techniques. The point-sampling used in each rendering pass is accurate and its simplicity allows the system to take advantage of any suitable polygon rendering hardware that may be available to speed up the process. The approach also allows the freedom to use any filtering technique that is appropriate to the cost and quality requirements of the particular application.

The main disadvantage of the method is that it does not produce an antialiased image in a single pass and indeed does not place any specific logical limit on the number of passes that may be required to produce an acceptable image. In its simplest form, it lacks any prescribed formal means of either assessing the quality of the current image, or of adaptively controlling its antialiasing effort to those portions of the image which may demand it most. While it appears that no solutions to this problem have been reported for the multi-pass method specifically, a considerable body of work has been done in the related field of non-uniform sampling. An ad-hoc adaptive non-uniform sampling technique has been reported for motion blur in a scanline renderer (Cook [1986]), but most other research in the area has been undertaken with ray tracing algorithms (Dippé and Wold [1985]; Lee et al. [1985]; Kajiya [1986]; Mitchell [1987]; Painter and Sloan [1989]; Mitchell [1991]), where there is considerably more scope for control over the distribution of samples on a pixel-by-pixel basis than with scanline or Z-buffer based approaches. Attempting to incorporate non-uniform sampling on a pixel-by-pixel basis in a polygon rendering system would considerably complicate the point-sampling procedure and thus obviate one of the main advantages of the multi-pass approach.

In the absence of such built-in control mechanisms, there appears little alternative to anecdotal recommendations about the number of passes required for an acceptable image: figures reported in the literature range from as few as 9 (Mammen [1989]) or 16 (Fuchs et al. [1985]; Barkans [1991]) for spatial antialiasing only, 23 passes for images exhibiting any one of motion blur, depth of field, or soft shadows, to as many as 66 for an image with motion blur, depth of field, soft shadows and spatial antialiasing as well (Haeberli and Akeley [1990]). Perhaps what this large variation demonstrates is that the number of passes that may be deemed necessary is highly dependent on the content of the image being rendered. However, the relative inefficiency of the approach in terms of the large number of passes that may be required to produce an image of sufficient subjective quality has been quoted as the reason for its rejection by some (Akeley [1993]).

5.12.1 Implementation of multi-pass antialiasing

Despite the practical problems mentioned in the preceeding paragraphs, multi-pass image accumulation remains an attractive antialiasing technique in an experimental environment. In the current implementation, it has allowed antialiasing to be added as an optional extra in the rendering pipeline, as an independent module which is fully compatible with the existing point-sampled polygon renderer. The various components of the multi-pass image accumulation system implemented are now briefly described.

The sampling method used to select subpixel sample point locations is non-uniform and stochastic in nature. It is based on an algorithm which generates a sequential approximation to a Poisson-disk sample distribution as described by Mitchell [1991]. For each antialiased rendering pass, a new subpixel sample point offset is selected. This subpixel offset is then conceptually applied to every sample point in the image, or equivalently the inverse offset is applied to every projected vertex in every primitive rendered. These offsets are applied between the perspective projection and scan conversion stages so that at the lower levels the point-sampling renderer is unaware that it is being used to sample at different subpixel positions.

Once a rendering pass is complete, the point-sampled image is filtered and then integrated into the image accumulation buffer. The filter currently used is a simple 1×1 pixel box filter, but wider filter supports (e.g. 3×3) or even different types of filter (such as Gaussian) may be used instead (Haeberli and Akeley [1990]). The accumulated image may be displayed after each pass by applying a simple intensity scaling operation which takes account of the number of samples accumulated so far before transferring the image to the frame buffer.

While there is no prescribed theoretical limit on the number of passes that may be accommodated into the filtered image, the relative precision used for the inten-

sity of each pixel in the sample and accumulated images determines an implicit upper bound on the number of samples that can be accumulated without overflow in practice. For example, if each sample rendering pass produces an 8 bit intensity value per colour channel per pixel and the accumulated image allows 16 bits per colour channel per pixel, then up to a maximum of $256 = 2^{16}/2^8$ passes may be integrated without overflow. An alternative strategy is to weight the contribution of each samples by a factor less than 1.0 (Haeberli and Akeley [1990]), although this results in some loss of precision from the sampled intensities.

5.12.2 Stereo view antialiasing

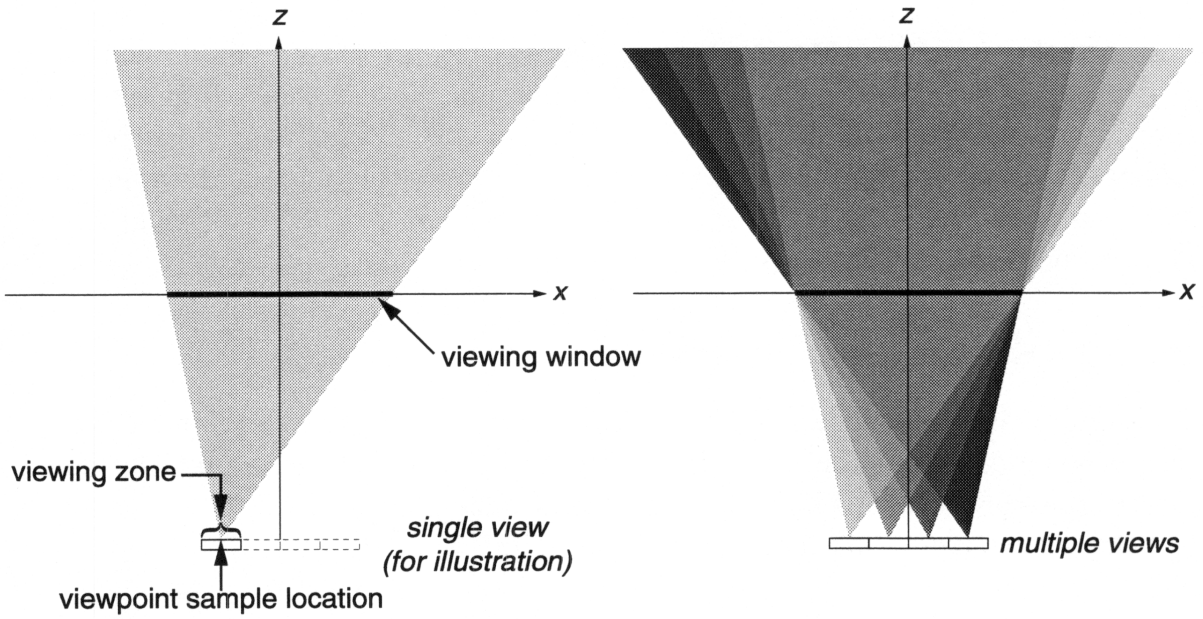
One of the main advantages of the multi-pass image accumulation technique is its flexibility and adaptability to a variety of image synthesis problems for which alternative solutions may be difficult to find, such as motion blur, depth of field, and soft shadows. Another application for image accumulation that is specific to a multi-view autostereo display is suggested by the mismatch between the discrete nature of the viewpoints used in the viewing model and the continuous nature of the physical viewing zones in which an observer can see each view of the image, as mentioned in Section 3.3.4.

The obvious approach is to place the viewpoint for each view in the middle of its corresponding viewing zone, as shown in Figure 5.11(a). An alternative approach is to distribute each viewpoint position across the horizontal extent of each viewing zone, as shown in Figure 5.11(b). A different viewpoint offset within each viewing zone is used for each pass just as different subpixel sample offsets are used with each pass for spatial antialiasing. Each view then represents an integration of what can be seen from across the whole viewing zone, rather than just from a single viewpoint.

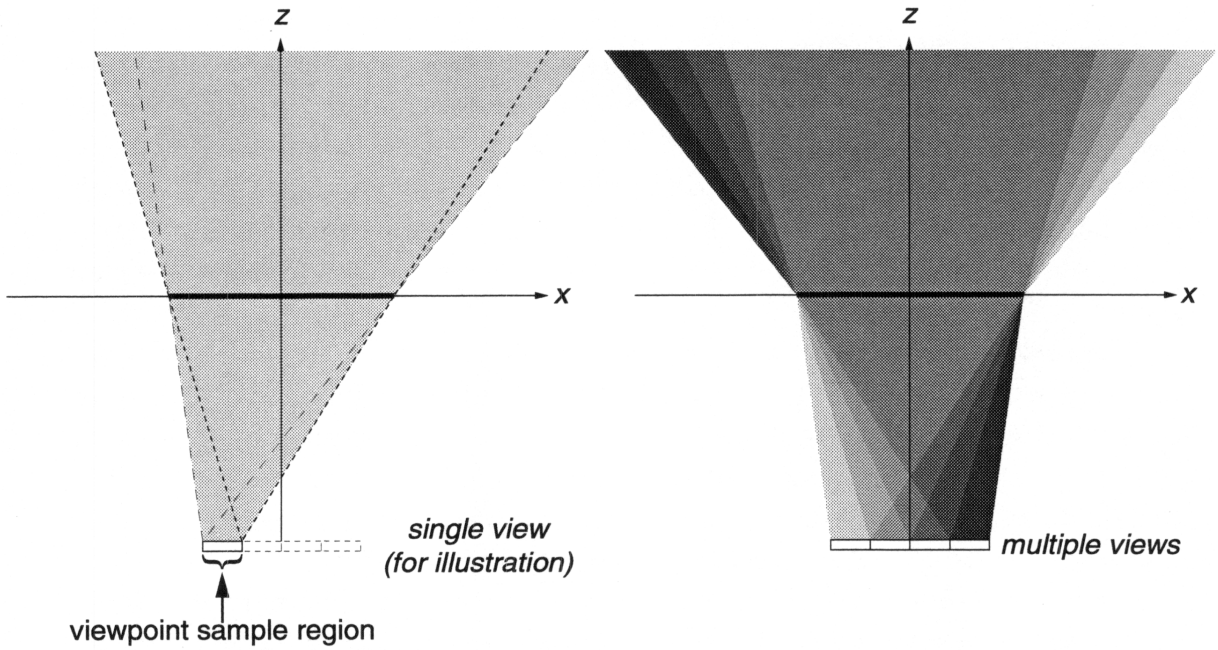
Dubbed *stereo viewpoint sampling*, this technique may be likened to using a camera with an aperture which is a horizontal slit rather than a pinhole. It produces images which exhibit blurring in the horizontal direction depending on the distance of the object from the viewing plane, in a manner similar to the way in which depth of field blurs objects in relation to their distance from the focal plane[2]. This effect is at its most noticeable with objects which exhibit large stereo disparities, which are precisely those objects which produce the most disturbing apparent jumping artifacts when the viewer moves his head from side to side. The subjective results of using this new sampling technique are discussed in Section 7.2.

It is possible to compare the effects of using stereo viewpoint sampling with those of temporal sampling in an animated sequence of images. Indeed, the views

[2]the difference being that depth of field blurs radially across the surface of a lens, whereas stereo viewpoint sampling blurs only horizontally across each viewing zone.



(a) viewpoint sampled at a single location in the middle of each viewing zone



(b) viewpoint sample locations distributed across entire width of each viewing zone

Figure 5.11: Viewpoint sampling within each viewing zone in a multi-view stereo image

of a stereo viewpoint sampled image are essentially equivalent to a sequence of motion-blurred images of a static scene taken at stereoscopically-related viewpoints with a moving camera, assuming that equal weights are applied to all temporal samples in each motion blurred image. Stereo viewpoint sampling may thus be considered a spatial stereo analogue of temporal motion blur.

5.13 Summary

This chapter has considered how to improve efficiency in a multi-view stereo renderer by taking advantage of the similarities between the views to reduce overheads and eliminate redundant processing. This is essentially achieved by factoring out the view-independent components of the process as much as possible, and sharing the results of such processing among all the views in the stereo image. It has been shown that this principle can be applied at almost all stages in the rendering pipeline, with the exception of the pixel-by-pixel visible surface processing of the Z-buffer. Chapter 6 examines alternative approximate rendering techniques for multi-view images which trade further speed improvements against a reduction in image quality.

6

Approximate Stereo Visibility

This chapter describes the motivation for and the implementation of an approximate visible surface algorithm for multi-view stereo images. First some existing related work is reviewed and its strengths and weaknesses discussed. The basic outline of the proposed approach is then presented, and some of the implementation issues are described.

6.1 Background and motivation

The Stereo Z-buffer algorithm described in Chapter 5 is a visible surface renderer optimised for multi-view stereo images. It is designed to produce images which are as accurate as those obtained from a conventional Z-buffer algorithm, yet to do so with greater efficiency than is possible simply by rendering each view independently of all the others. While much use of stereo coherence is made at a variety of different levels in this algorithm, comparatively little is applied to the visible surface determination step itself. Yet there is still much in common between the views even at this level which remains to be exploited by a visible surface determination algorithm.

This idea of course is far from new. A number of reported approaches to multi-view visible surface determination (stereo or otherwise) were identified in Section 3.2.3. Some of these approaches will be examined here in a little more detail and their principal features discussed.

6.1.1 Previous work

Computer generated holographic stereograms share many similarities with multi-view autostereo images. Also known as *stereograms* (or *integrams*), they are made from a number of individual images recorded from multiple points of view along a straight or curved path (Naimark [1991]). One method for rapidly generating such images is described by Guo et al. [1988]. It works by first constructing an intermediate point-sampled three-dimensional approximation of the original scene in a so-called "Position Oriented" (PO) buffer. The points in the PO buffer are obtained using a process similar to conventional ray tracing to sample the scene along parallel rays in a given viewing direction, but independent of the perspective of any particular viewpoint. Unlike conventional ray tracing however, information about all the intersections of each ray is retained in the PO buffer, not just the intersection nearest to the origin of the ray. To produce each view, the appropriate viewing transformation is applied to each point in the PO buffer, followed by perspective projection. The visibility of overlapping points in each view is resolved using depth comparisons at each pixel in a manner similar to a conventional Z-buffer.

Artifacts such as unwanted gaps in the reprojected views were anticipated and two strategies for reducing them are suggested by Guo et al. [1988]. The first is to increase the number of sample points in the PO buffer, either by increasing the sampling density, or by sampling in more than one viewing direction. The second involves applying an ad-hoc postfiltering algorithm which attempts to detect and correct errant pixels in each of the reprojected views. No objective measures of the effectiveness of these corrective strategies is provided in the paper, although the high cost of increasing the number of samples in the PO buffer is acknowledged.

Most other researchers concerned with the problem of how to share visible surface information between a number of views have been interested in animated images. An important observation about the nature of the multi-view visible surface problem for stereo images is its similarity to the same problem for animated sequences of images. Indeed, a static multi-view stereo image can be thought of as an animated sequence of images where the camera moves in relation to a stationary scene, and this is where the two problems converge.

The similarity between an animated sequence of images and the views of a stereo image has lead to some of the most promising research in the field of stereo image generation in recent years. Ezell and Hodges [1990] present an algorithm for reducing the cost of ray tracing the second view of a stereo image pair based on a technique originally applied to sequential frames of animation (Badt [1988]). The algorithm works by directly reprojecting each pixel from a fully ray traced left eye view to an approximated right eye view, according to a stereoscopic perspective relationship similar to that described in Section 4.2.3. The reprojected view was ex-

pected to exhibit a variety of artifacts — such as missed, overlapped, or "bad" pixels — which an ad-hoc postprocessing clean up stage was intended to detect, in a manner not unlike that described by Guo et al. [1988]. Any errant pixels found in the reprojected view were then to be corrected by tracing new rays through them.

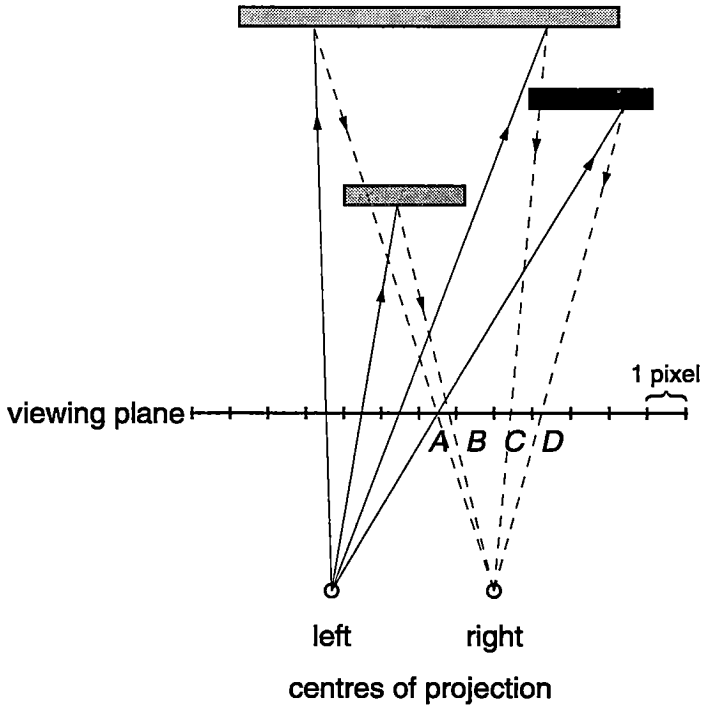


Figure 6.1: Problems in reprojected images. Pixel A is hit by more than one reprojected ray. B is missed entirely by any reprojected rays. C has a ray reprojected onto it, but it is not from the surface which should be visible at that point. D is reprojected without problems.

The problems of missing, overlapped, and bad pixels are illustrated in Figure 6.1. Pixels on the viewing plane sampled from the left centre of projection in the left eye view through the centre of each pixel are reprojected onto the viewing plane through the right centre of projection for the right eye view. Pixel A is hit by two reprojected pixels and is thus classed as an *overlapped* pixel, where the correct visible pixel is found by comparing the depths of the overlapping pixels. Pixel B is not hit at all in the reprojected view and is thus classed as a *missing* pixel, requiring retracing. Pixel C is hit by a pixel reprojected from a surface which should not be visible at that point in the reprojected view, and is thus classed as a *bad* pixel. While missing and overlapped pixels can be easily recognised and readily dealt with, the heuristic methods described by Ezell and Hodges [1990] fail to reliably identify bad pixels in the reprojected view.

A solution to these difficulties was suggested by Adelson and Hodges [1993], whereby the algorithm described by Ezell and Hodges [1990] was modified to use a more conservative and reliable bad pixel detection and correction strategy. Furthermore, the basic reprojection algorithm was adapted to cope with antialiasing using multiple sample rays per pixel, although the results are strictly correct only if filtering is performed following reprojection. Attempts to extend the technique beyond first-level visible surface rays were largely unsuccessful however, and all subsequent reflection or refraction rays were traced independently for each view. Nonetheless, it was claimed that the technique is always at least as efficient as tracing the two views separately, and usually much more efficient, especially with scenes containing few reflective or refractive elements.

Although the algorithms of Guo et al. [1988], Ezell and Hodges [1990] and Adelson and Hodges [1993] are based on ray tracing techniques, there is no reason why a similar reprojection strategy may not be used with a Z-buffer visible surface algorithm, as outlined by Harrison and McAllister [1993]. Using a complete Z-buffered view as its starting point, it works by applying a z-shear transformation to each pixel to obtain the position of the pixel in the reprojected view. Although the existence of unwanted gaps in the reprojected view is acknowledged, it is argued (somewhat unsatisfactorily) that this is not necessarily a significant problem, and thus few indications are given about how it might be addressed. However, it is suggested that for a multi-view panoramic stereo image, higher quality may be obtained by increasing the number of views computed exactly, with any remaining intermediate views produced by interpolation.

In an entirely different context, a generalisation of this reprojection technique was presented by Chen and Williams [1993]. The idea is to apply image morphing techniques to three-dimensional projected views of a scene to enable intermediate views to be synthesised. Potential applications of the technique include walkthroughs in virtual environments and virtual holograms, although it is sufficiently general to be used in a variety of other situations as well. The algorithm works by automatically computing pixel-by-pixel correspondences between two (or more) views and their associated depth maps (Z-buffers) using knowledge of the camera movement between the views to obtain the appropriate transformations. The set of these pixel correspondences constitute three-dimensional "morph maps" between the views, which may then be used to interpolate where the pixels should appear as seen from any given viewpoint between the original views. Optimisations described include applying quadtree block compression to the morph maps, so that entire blocks of pixels at similar depths can be reprojected with each interpolation operation rather than just an individual pixel.

However, this forward mapping strategy suffers from essentially the same problems of overlaps and holes in the interpolated image described (in slightly different

terms) by Ezell and Hodges [1990]. Chen and Williams [1993] resolve overlaps using depth priorities, and estimate the intensities of any missing pixels using interpolation between their neighbours as an alternative to ray tracing. As before, it is the bad pixels that are the most difficult to detect. Chen and Williams [1993] suggest that this problem may be alleviated (but not necessarily eliminated) by increasing the number of source images used in the interpolation, so as to reduce the size of the ambiguous regions which are seen in none of the source views yet should appear in an intermediate view.

A similar approach to view interpolation was used by Ward [1994] to reduce the cost of generating sequences of images of static environments for animated walkthroughs. This algorithm uses a sample image and its corresponding depth map as an approximate three-dimensional model of the visible surfaces in a scene from a particular viewpoint. Information about the camera transformations between the views are used to reproject pixels from the original view as they would appear from an arbitrary new viewpoint. A Z-buffer is used to resolve overlaps in the reprojected view, with any errant pixels detected being either interpolated from neighbouring pixels or re-traced using the original scene description. For relatively large camera movements or significant changes in visibility, multiple sample images and depth maps may be used to reduce the number of errors in the reprojected view.

6.1.2 Problems with reprojection

All the algorithms outlined in Section 6.1.1 use reprojection of a relatively small amount of known image information to obtain an approximation of an unknown view, rather than applying a visible surface process to the entire original database description to obtain an exact solution. Artifacts in the reprojected view are traded off against the reduction in overall rendering cost made possible by using this approximation. Depending on the severity of the artifacts and the intended application, attempts may be made to detect and correct these errors where it is possible and considered efficient to do so. However image reprojection as a technique has some problems, particularly when used to generate stereo images.

One problem is that reprojection of a point sampled image does not necessarily produce the same results as point sampling reprojected geometry, as discussed by Adelson and Hodges [1993]. It is due to the loss of information inherent in the point sampling process, which only gives an approximation to the correct geometry. An example best illustrates the nature of the phenomenon.

Figure 6.2(a) shows a single scanline of an image with a span from a typical polygon superimposed on it. The heavy lines indicate the actual edge of the polygon as obtained from the projection of its geometry, while the shaded squares indicate the pixels which are considered to fall within the span by point sampling at the cen-

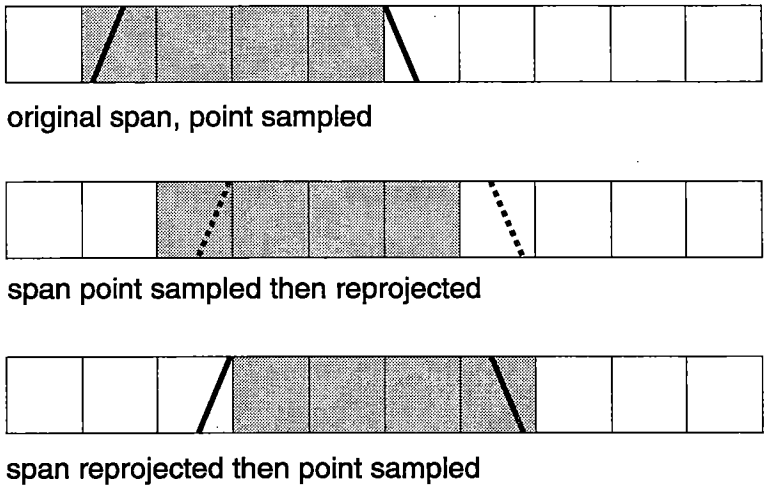


Figure 6.2: Reprojection of a polygon span

tre of each pixel. Note that there is a discrepancy between the intersections of the span edges with the horizontal scanline and the centres of the pixels considered to represent each end of the span, inherent in the point sampling.

If we assume that the polygon represented in Figure 6.2 is parallel to the viewing plane, then all points in the polygon share the same Z value and thus will be reprojected by the same amount in the neighbouring stereo view. Figure 6.2(b) shows the result of reprojecting the point sampled span from Figure 6.2(a) by 1.4 pixels, where the dotted lines indicate where the reprojected edges of the polygon should appear. In contrast, Figure 6.2(c) shows the result of point sampling only after reprojecting the original span geometry from Figure 6.2(a) by the same amount. Differences can clearly be seen at both ends of the span in Figure 6.2(b) when compared to the correct span shown in Figure 6.2(c). These discrepancies produce a change in the observed parallax between the original and reprojected spans which may be perceived as an apparent difference in the depth of the span when viewed in stereo.

If the polygon is not parallel to the viewing plane, then different points in the polygon will have different depths and thus will be reprojected by different amounts in the neighbouring stereo view. If the intersection of the span edges with the scanline have different depths, then these depths will themselves differ from those of the pixels at the ends of the span obtained by point sampling. As the depths differ, so the stereo parallax will differ, and thus the endpoints of the span which is reprojected after point sampling will not be shifted by the same amount as the endpoints of the span which is point sampled only after reprojection. Although stereo parallax changes relatively slowly as a function of depth and thus significant parallax errors at span endpoints may only occur with relatively large depth slopes,

even a single pixel error in stereo parallax may produce a relatively large error in perceived depth, as illustrated in Figure 3.9.

Another problem with reprojection is that it cannot generally be used with antialiased source views, owing to the view dependent nature of the antialiasing operation (Chen and Williams [1993]). Each pixel in an antialiased image may have intensity contributions sampled from a number of different objects in the scene, and each object may be at a different depth. This is particularly troublesome around object silhouettes, where both depth values and intensities may vary considerably among the samples affecting a given pixel. It is therefore not generally possible to find a unique reprojection of an antialiased pixel as seen from a different viewpoint, as each sampled component of the antialiased pixel should ideally reproject to a different position in the alternate view. Attempting to reproject an antialiased silhouette pixel based only a single depth value causes an unwanted ghost outline image to appear at the reprojected depth, corresponding to all those samples with depths differing from that used in the reprojection (Adelson and Hodges [1993]). The only reliable way to avoid this problem is to reproject each sample at high resolution before applying any antialiasing filter, effectively implementing a supersampling technique (Chen and Williams [1993]; Adelson and Hodges [1993]).

6.1.3 Motivation for a new approach

Taking into consideration the strengths and weaknesses of the algorithms outlined in Section 6.1.1, there are a number of areas where opportunities for improvements may be found. For high-quality approximations, the accuracy problems introduced by reprojecting a point sampled image (as mentioned in Section 6.1.2) may be unacceptable. If antialiasing is required, then reprojection must be done at high resolution before filtering down to the desired resolution (Chen and Williams [1993]), and thus the reprojection operation must be an integral part of the whole image generation process. This effectively precludes purely image-based reprojection such as those described by Chen and Williams [1993] and Harrison and McAllister [1993].

Further opportunities may present themselves if reprojection is closely tied in with the rendering system. One example of this is the way in which errant pixels in the reprojected view may be corrected by selective ray tracing of the original database, as used by Ezell and Hodges [1990], Adelson and Hodges [1993], and Ward [1994]. The question of whether similar opportunities may exist in a Z-buffer based renderer provides the motivation for the approach described in Section 6.2. For example, if a ray tracing algorithm reprojects at the level of each pixel sample, a Z-buffered polygon renderer may choose to reproject at the level of each polygon. It may then be possible to reduce errors in the reprojected view by rendering selected polygons from the original database.

Other possibilities arise when applying reprojection techniques to multi-view stereo images. Accelerated rendering techniques for two-view stereo images have been inherently limited to fully sampling one view to infer probable visibility in the other (Ezell and Hodges [1990]; Adelson and Hodges [1993]). For multi-view stereo images, the artifacts caused by gaps in the interpolated views may be alleviated by the use of additional sample views, in a manner similar to that suggested by Guo et al. [1988], Harrison and McAllister [1993], Chen and Williams [1993], and Ward [1994]. Thus the number of sampled views may be used as a parameter to trade off the overall quality of the resulting image against its rendering cost.

6.2 The approximate stereo visibility algorithm

This section describes the proposed approximate stereo visible surface algorithm, both at a conceptual level and in terms of the implementation issues involved. Where appropriate, the proposed algorithm is compared with some of the previous similar techniques outlined in Section 6.1.1. Details of the actual implementation of the algorithm are given in Section 6.3. Practical comparisons of the observed performance of the proposed algorithm with the Stereo Z-buffer are made in Chapter 7.

6.2.1 Outline of the algorithm

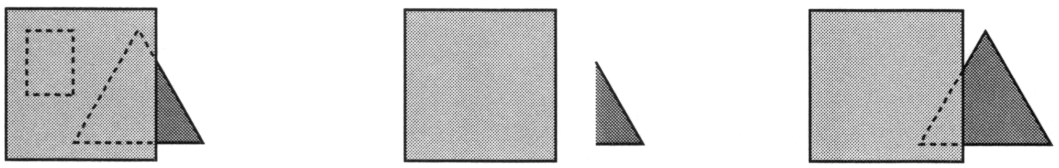
The guiding principle behind the proposed approximate stereo visible surface algorithm is that of *visibility coherence*. That is, if an element of the scene is visible in a given view of a multi-view stereo image, then it is likely that the same element is also visible in other nearby views of the same image. This is the same principle which implicitly underlies almost all the reprojection algorithms[1] described in Section 6.1.1. Previously however, it has generally only been applied at the level of individual pixel sample elements: the idea proposed here is to investigate what (if any) benefits may be obtained if the definition of a visible element is extended to cover larger, higher level image elements.

A similar idea was used by Chen and Williams [1993], where applying quadtree compression to the morph maps allows blocks of pixels to be reprojected as a single unit. With the proposed method, the same principle is applied to logical elements in the scene rather than physical groups of pixels and thus the nature of any artifacts in the reprojected view is expected to be in terms of these logical elements instead of physical pixels. By retaining the physical *cohesion* of the primitive element used

[1] only Guo et al. [1988] do not use visibility information to guide the reprojection effort

as the unit of reprojection, pixel gaps within any given primitive element cannot appear in the reprojected view. Thus the new strategy may be expected to significantly reduce the number of errant pixels classed as bad in previous reprojection algorithms. Implicit in this approach is the way in which reprojection is performed prior to sampling, thus avoiding the sampling accuracy problems mentioned in Section 6.1.2.

An example of the kind of benefit this strategy may provide is illustrated in Figure 6.3. A simple scene is shown consisting of a square, a triangle, and a rectangle. The square in the foreground partially occludes the triangle some distance behind it, and completely obscures the rectangle in the background. Figure 6.3(a) shows the scene as it appears in the original sample view. Figure 6.3(b) shows the same scene with all visible *pixels* reprojected to the right, as may be produced using an image-based reprojection method. Note how a gap appears in the reprojected view where the portion of the triangle hidden in the sample view (a) should be found. Figure 6.3(c) shows the same scene with all visible *polygons* reprojected to the right using the proposed strategy.



(a) original sample view

(b) view of reprojected visible *pixels*

(c) view of reprojected visible *polygons* (proposed strategy)

Figure 6.3: An example of the benefits of the proposed reprojection strategy. (a) shows the original sample view, with hidden portions indicated using a dotted outline. (b) shows the reprojection of visible *pixels* from (a). (c) shows the reprojection of visible *polygons* according to the proposed strategy.

It can be seen that the portion of the triangle hidden in the original sample view (Figure 6.3(a)) is recovered correctly in the reprojected view using this new polygon-based reprojection technique (Figure 6.3(c)), in contrast to the image-based approach (Figure 6.3(b)). If the triangle was shaded or texture mapped, this information too could be recovered in the reprojected view using the new method. However, not all the triangle is visible in the reprojected view: that part which remains hidden by the square must be eliminated by conventional visible surface techniques such as a Z-buffer, which contributes to the cost overheads of the new technique. The rectangle is not reprojected at all, as it is not visible in the original sample view. This provides

an important computational saving that is also used by almost all the reprojection algorithms described in Section 6.1.1.

The same basic approach can in principle be applied at any logical level in the scene description hierarchy. For example, instead of reprojecting visible polygons, it might be useful to reproject groups of connected polygons which are structured to form objects in the scene. Taking it a step further, groups of related objects in the same spatial neighbourhood might be used as the unit of reprojection. Each additional level up the scene hierarchy increases the potential benefits in terms of the quality of the reprojected view, but also increases the associated reprojection cost. Thus one of the most critical decisions to be made when implementing a system based on this strategy is to determine what element level will be used as the unit of reprojection in order to obtain a good quality/cost trade off. The key to the success of the proposed technique is how good the visibility of a given element in one view is at predicting its visibility in a neighbouring view.

How efficiently the visibility of any given element is determined is another important consideration with the proposed method. Finding the precise visibility of a particular element may prove too costly in general, or simply be inconvenient to accommodate within the framework of the visible surface algorithm used. In any case, exact visibility measures may not be required, and less expensive approximations may suffice for the strategy to work well enough. These and other implementation issues are examined in more detail in Section 6.2.2.

6.2.2 Implementation issues

As discussed in Section 6.2.1, the major implementation issues associated with the proposed approximate stereo visibility algorithm are:

1. how efficiently visibility is estimated, and

2. at what logical level in the scene the conditional reprojection is applied.

The method used to determine the visibility of an element in the sample view is crucial to the efficiency of the proposed approach. While some algorithms are capable of providing precise visibility information for an individual scene element in a manner which is readily integrated as each element is processed, other algorithms may be less accommodating. For example, a scanline visible surface algorithm is designed to determine precisely which element is visible at each pixel in the view as that pixel is processed. This makes it well suited to the task of providing exact

visibility information at the level of each pixel. In contrast, it is not generally possible for a Z-buffer algorithm to make valid claims about the definitive visibility of any elements in the view until all elements have been processed, as one of the strengths of a conventional Z-buffer is that it allows elements to be processed in any order. Thus an element which is found to be visible by a Z-buffer at an intermediate stage in the rendering of the scene may subsequently be overwritten by other nearer elements in the scene, depending on the relative order in which the elements are processed. If precise visibility information is required, this may mean that visible surface determination must be completed in all sample views for all elements in the scene before any reprojection is performed, thus requiring two distinct passes at the highest level in the rendering process.

However, it may not be necessary to be so rigid in the requirements placed on the visible surface algorithm for the approach to be useful. For example, if the precise visibility requirement is relaxed, then even a simple Z-buffer could be used to estimate the visibility of an element as it is being rendered in the sample views. Assuming elements are rendered in a random order, this strategy might be expected to result in about half the elements being subsequently overwritten by closer ones on average. This means that about half of all elements initially considered visible in the sample views may be unnecessarily reprojected in the approximate views, which is clearly inefficient. If however elements are rendered in front to back order, then a considerably lower proportion of the elements may reasonably be expected to be redundantly reprojected in this manner.

The way in which the visibility of a given element is dependent on the visibility of its component parts gives rise to issues concerning the level at which reprojection is applied. In order to determine visibility information at the level of an arbitrary scene element, it is necessary to take into account the visibility of all lower level elements that make up the image of that entity. The approximate visibility algorithm thus requires a two pass approach: render the element in the sample view to obtain an estimate of its visibility, then if visible render the element in the reprojected view. The problem with such a two-step process is that it may not be possible to take the fullest advantage of the coherence between the stereo views during the rendering procedure. For example, if the stereo Z-buffer algorithm described in Chapter 5 is used for visible surface determination, and conditional reprojection is based on visibility at the polygon level, then the stereo polygon scan conversion procedure described in Section 5.9 will have to be applied twice for each polygon with consequent duplication of effort. If however conditional reprojection is based on the visibility of each horizontal span in the polygon, most of the advantages of stereo polygon scan conversion can be maintained.

6.3 Implementation

The current implementation of the Approximate Stereo Visibility algorithm obtains visibility estimates using a Z-buffer. Primitives are first bucket sorted by depth, and the buckets subsequently processed in order from those nearest the viewpoint to those most distant. This approximates the desired front-to-back depth order, and allows the Z-buffer to provide useful visibility estimates as each primitive is processed. Although the explicit depth sort does have the disadvantage of placing additional storage demands on the algorithm, the bucket sort is relatively inexpensive in computational terms, having a growth in computational complexity which is linear in the number of primitives processed.

In order to maintain the coherence advantages of the single-pass stereo scan conversion technique, visibility is decided on the basis of individual horizontal spans within the polygon, rather than at the polygon level itself. Reprojection thus takes place within the inner loop of the stereo scan conversion procedure, which allows almost all the computational benefits of utilising stereo coherence between the views to be retained. Reprojection at the span level also maintains a degree of physical cohesion which prevents many bad pixel errors from ever appearing in the reprojected views. Some degradation of quality in the reprojected view is expected by using span (instead of polygon) visibility in this manner, but much is gained in terms of simplicity.

A span is rendered in each of the sample views in turn, until either it is found to be visible in one of them or all sample views have been examined. If it is found to be visible in one of the sample views, it is assumed to be likely to be visible in other views as well, and so is rendered in each of the remaining views in the image. If it is not visible in any of the sample views, it is assumed to be unlikely to be visible in any of the other views and is thus discarded.

This process is illustrated in Figure 6.4 for an n- view stereo image. The two outermost views (V_1 and V_n) are selected as the sample views; the remaining views (V_2 to V_{n-1}) will be approximated. The three polygons shown are rendered in front-to-back order: A, B, C. Polygon A is rendered first in sample view V_1 and immediately found to be visible, so it is rendered in all remaining views (V_2 to V_n). Polygon B is not visible in sample view V_1 but is (partially) visible in sample view V_n, so it is rendered in all remaining views (V_2 to V_{n-1}). Polygon C is not visible in either sample view V_1 or sample view V_n, and thus is discarded without being rendered in any of the approximated views (V_2 to V_{n-1}) at all.

This simple approach only works if the visibility of a polygon changes at most once between sample viewpoints. An example of this problem is shown in Figure 6.5. Polygon B is visible in neither sample view V_1 nor sample view V_n and thus is not rendered in any of the intervening views (V_2 to V_{n-1}), despite the fact

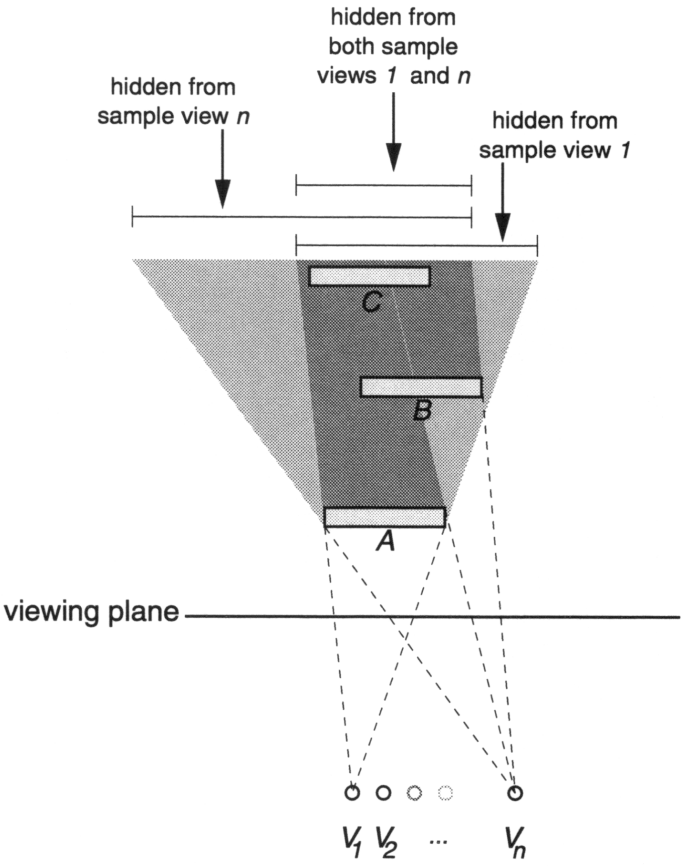


Figure 6.4: An example of how Approximate Stereo Visibility works. Polygons A, B, C are processed in front-to-back order. Only if a polygon is visible in one of the sample views (V_1 or V_n) is it drawn in the remaining views in the image.

that it should be visible in view V_i. While it is difficult to detect such situations in general, it is possible to reduce the probability of their occurrence by using additional sample views to increase the visibility sampling density. A similar suggestion was made by Chen and Williams [1993] to alleviate the problems of holes in the image which result from regions of the scene that are not represented in any sample views. The disadvantage of using additional sample views is that rejecting genuinely invisible polygons (such as C in Figure 6.4) becomes more costly, as a polygon is discarded only when it has been shown to be *not* visible in *all* the sample views. This provides an opportunity to trade off the cost of increasing the number of sample views against the decreased error frequency (and thus improved overall image quality) in the approximate views.

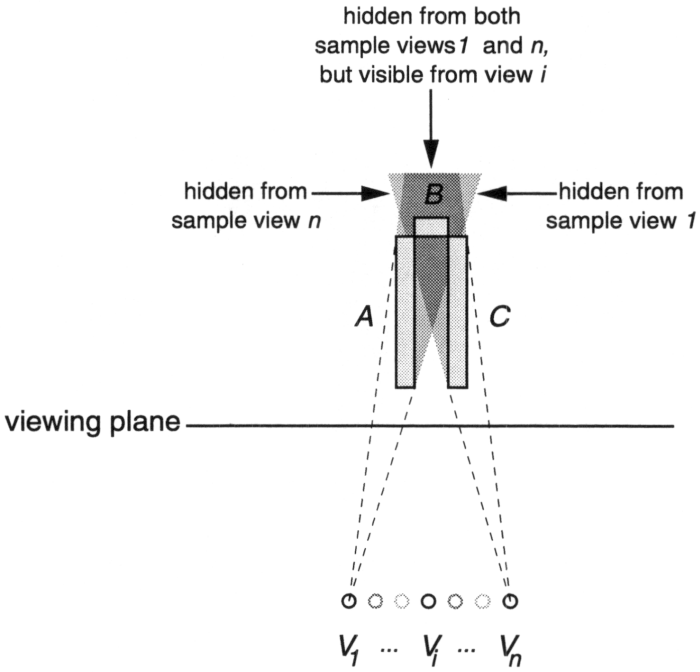


Figure 6.5: Where Approximate Stereo Visibility may fail. B is not drawn as it is not visible in either of the sample views V_1 or V_n, despite the fact that it should be visible in view V_i.

6.4 Summary

In this chapter, existing approximate rendering techniques for multi-view images have been adapted to a multi-view stereo Z-buffer system. This effectively allows visible surface information to be shared between the views in a multi-view stereo image, in a way that is not possible with the Stereo Z-buffer algorithm described in Chapter 5. Although the approximate nature of the technique does mean that some degradation in image quality is expected, it is possible to control this to some extent by trading off improvements in rendering speed against the probability of errors in the image. An experimental comparison of the Stereo Z-buffer with the Approximate Stereo Visibility algorithm is presented in Chapter 7.

7

Evaluation of stereo rendering

This chapter is concerned with the evaluation and comparison of an implementation of the stereo rendering techniques described in Chapters 5 and 6. The observed performance of each method in terms of both objective measurements and subjective image quality is presented, and comparisons are made between the two approaches.

7.1 Objective performance

The objective performance of the two rendering algorithms was measured experimentally by finding the total time taken by each algorithm to render a test scene under a variety of different conditions. The actual observed performance data is presented in Sections 7.1.1 and 7.1.2. First however the test scene database and other conditions common to the tests used with both algorithms are described.

An imaginary city tower block landscape was used for the test scene database. It consists of a 9×9 grid of city blocks, where each block is made up of a 3×3 grid of simple building objects with randomly selected heights. Each building object is made from four wall polygons and one roof polygon, with a floor polygon shared among all buildings in a block. The test database thus contains a total of 3726 four-sided polygons. Each polygon was treated as an independent entity by the rendering process, making no use of topological properties (such as shared vertices be-

tween polygons in each building) to reduce computational costs.

The objects in this test database are clustered near the ground plane and thus have a non-uniform distribution in three-dimensional space. The apparent depth complexity of the scene may therefore vary, depending on the vantage point and viewing direction. Depth complexity is at a maximum when viewing parallel to the ground plane, and at a minimum when viewing perpendicular to the ground plane. This is convenient for testing purposes as it allows scenes of different depth complexities to be rendered using the same underlying database, thus maintaining the same computational load for viewpoint-independent processing such as database traversal and object transformations. The test database was rendered from five different vantage points at various elevations with respect to the ground plane of the database, as illustrated in Figure 7.1.

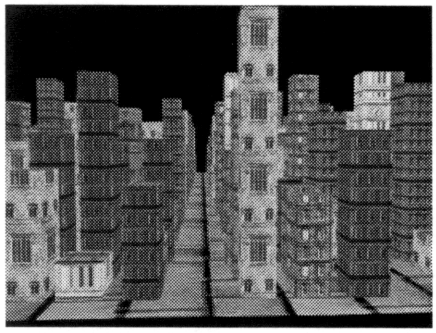
Three different shading methods were tested for each scene, to allow their relative costs to be compared. Flat shading applies a constant intensity to each pixel in a given polygon and is the simplest method. Smooth shading varies the intensity across the interior of the polygon by interpolating shading parameters associated with each vertex of the polygon. Texture mapping projects an arbitrary image onto the surface of the polygon, where texture reference coordinates interpolated between the vertices of the polygon are used to determine which pixel in the texture map image maps to each pixel in the polygon. Even with simple point sampling of the texture map rather than proper filtering techniques, this is the most computationally expensive shading method of the three tested. Smooth shading and texture mapping are discussed in more detail in Section 5.11.

A 320×240 monochrome (8 bits per pixel) display mode was used in all rendering tests (format L16 from Table 2.1). This allowed the greatest maximum number of views (16) to be used for the stereo image. The actual number of views rendered was varied as part of the test procedure in order to demonstrate the relationship between the number of views in the stereo image and the observed performance. The tests were performed using the experimental computer graphics platform described in Section 2.5, with all code written in the C programming language (Kernighan and Ritchie [1988]).

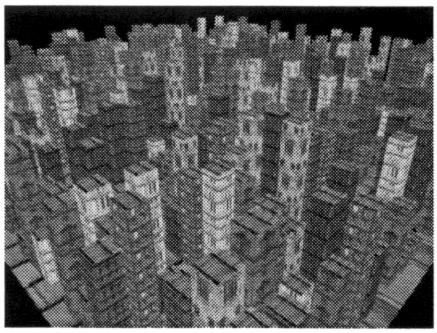
7.1.1 Stereo Z-buffer

In this implementation, shading is only performed for a given pixel if the intermediate results of the Z-buffer algorithm indicate that the object is visible at that pixel. For a scene with many overlapping objects (and thus high average depth complexity), the order in which objects are processed can have a considerable impact on how much shading is performed. The more expensive the shading calculation, the greater the impact of depth ordering on the observed performance. To allow this

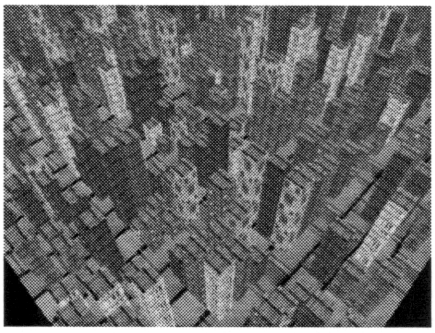
Scene 1



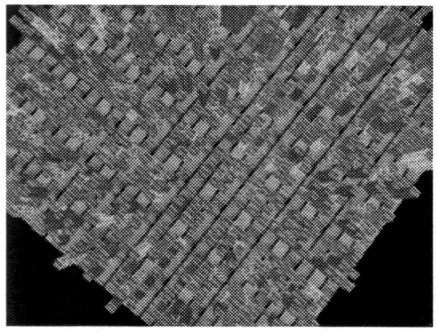
Scene 2



Scene 3



Scene 4



Scene 5

Figure 7.1: The test database as seen from each of the five vantage points

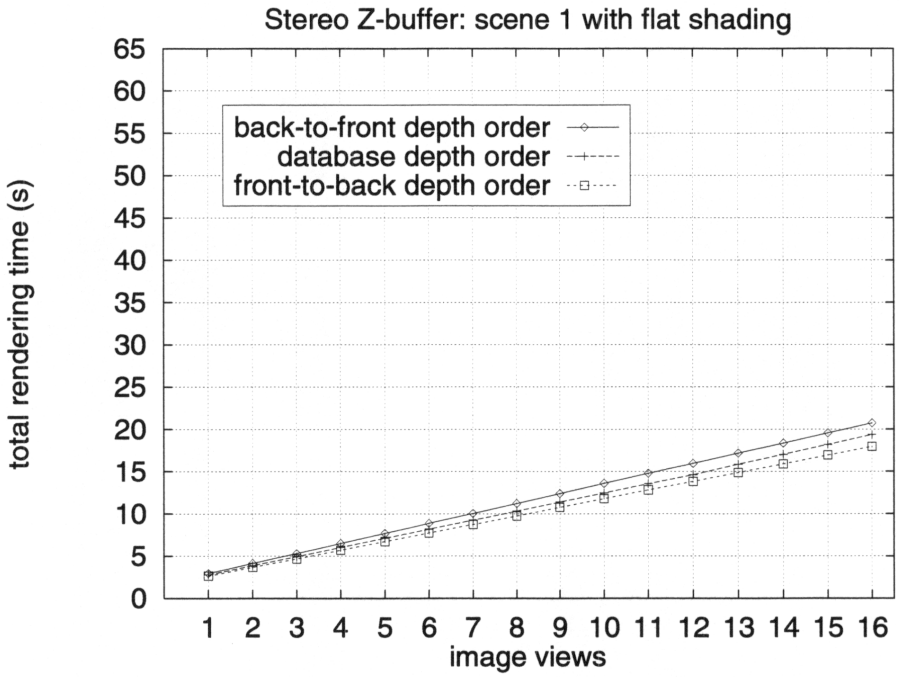


Figure 7.2: Observed performance of the Stereo Z-buffer for scene 1 with flat shading

effect to be quantified, primitives from the scene database were processed in three different depth orderings in separate tests — front-to-back, back-to-front, and natural database order — corresponding to the expected best, worst, and (nominal) average cases respectively.

For front-to-back order, primitives are first bucket sorted by nearest depth from the observer, then the contents of each bucket are processed starting from the nearest bucket to the most distant. Note that this only produces an approximation of the correct ordering as there is no discrimination of depths within each bucket, nor any consideration of the relationship between any given pair of primitives to determine whether one actually appears in front of the other. A similar technique was used for back-to-front order, with primitives first bucket sorted by farthest depth and buckets processed from most distant to nearest. For database depth order, no explicit depth ordering is performed and primitives are processed in the order they are presented to the visible surface algorithm. To ensure both a good utilisation of the available Z-buffer resolution and a reasonable spread throughout the range of depth buckets for the ordered modes, the Z-values are scaled to fit the minimum and maximum depth values of all primitives present in each scene.

The observed rendering performance of the Stereo Z-buffer for scene 1 is illus-

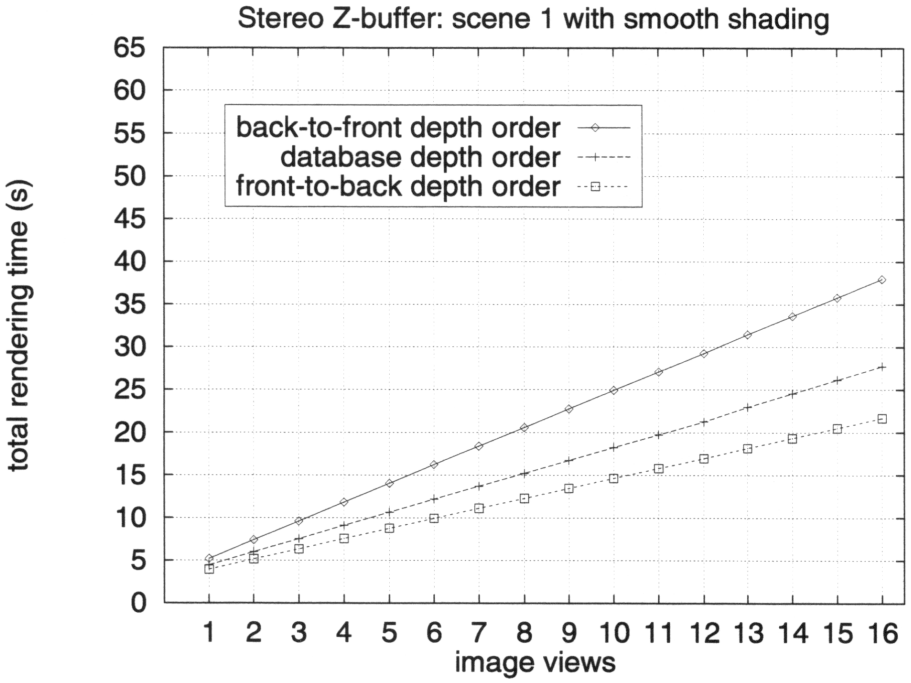


Figure 7.3: Observed performance of the Stereo Z-buffer for scene 1 with smooth shading

trated in the graphs of Figures 7.2, 7.3 and 7.4 for flat shaded, smooth shaded, and texture mapped polygons respectively. In each graph, the number of image views rendered is shown along the horizontal axis and the observed rendering time is shown on the vertical axis. Three separate traces are shown in each graph, one for each depth ordering (front-to-back, back-to-front, and database order).

There are several observations which may be made about the data shown in the graphs of Figures 7.2 to 7.4. The first is that the relationship between between the number of rendered image views and the total rendering time appears to be a linear one. The second is that both the depth ordering and the shading method can have a significant effect on the total rendering time.

Similar observations may be made about the performance of the Stereo Z-buffer when applied to scenes 2 to 5. This suggests a simple characterisation of the total rendering time t_r in terms of the rendering time per image view t_v and the number of views rendered n:

$$t_r = n \cdot t_v + t_o \tag{7.1}$$

where t_o is the overhead processing time associated with all parts of the rendering pipeline that are not directly involved with view-dependent computations.

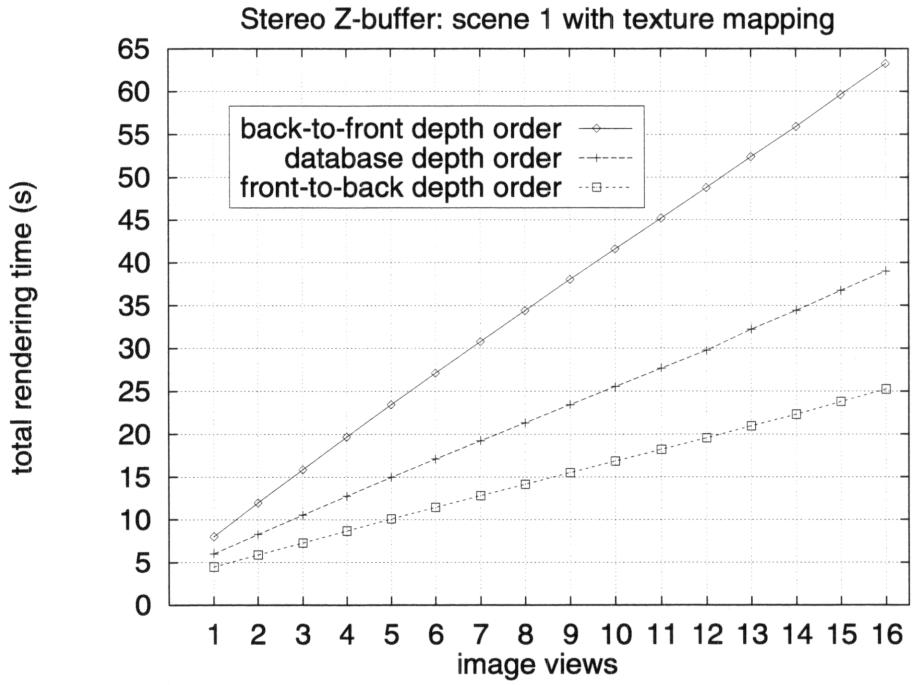


Figure 7.4: Observed performance of the Stereo Z-buffer for scene 1 with texture mapping

Note that both t_v and t_o may depend on the depth order, shading method, scene and database used.

If we apply linear regression techniques to the rendering performance data for scenes 1–5, it is possible to obtain empirical estimates of t_v and t_o for each combination of depth order, shading type, and scene used with the test database. The results of such an analysis for t_v are shown in Figure 7.5, which summarises the observed performance of the Stereo Z-buffer in terms of the rendering time per image view using database depth order for each scene with flat shading, smooth shading and texture mapping. Error bars denote the variation in observed rendering time per image view when primitives are processed by the Z-buffer in best (front-to-back) and worst (back-to-front) depth ordering.

From Figure 7.5 it is clear that the observed rendering performance of the Stereo Z-buffer varies substantially with each scene, shading type and depth ordering. The performance differences observed between the scenes were expected, and are due to differences in the visible content of each scene. All other things being equal, the variation in performance with shading type also follows the predicted pattern, with flat shading the least expensive and texture mapping the most costly of the three methods tested. However, the variability of performance with depth order-

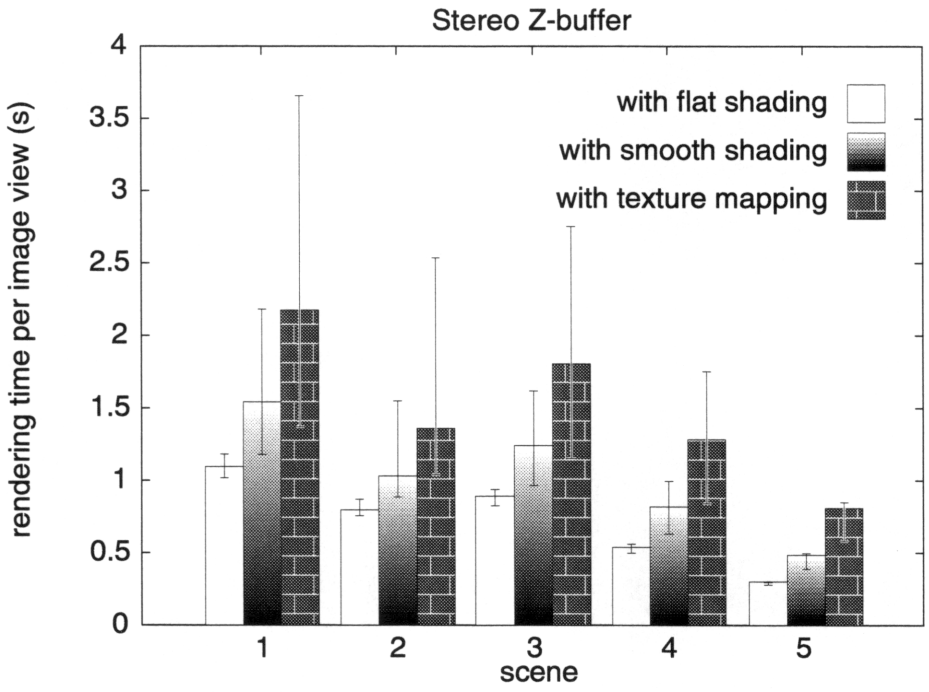
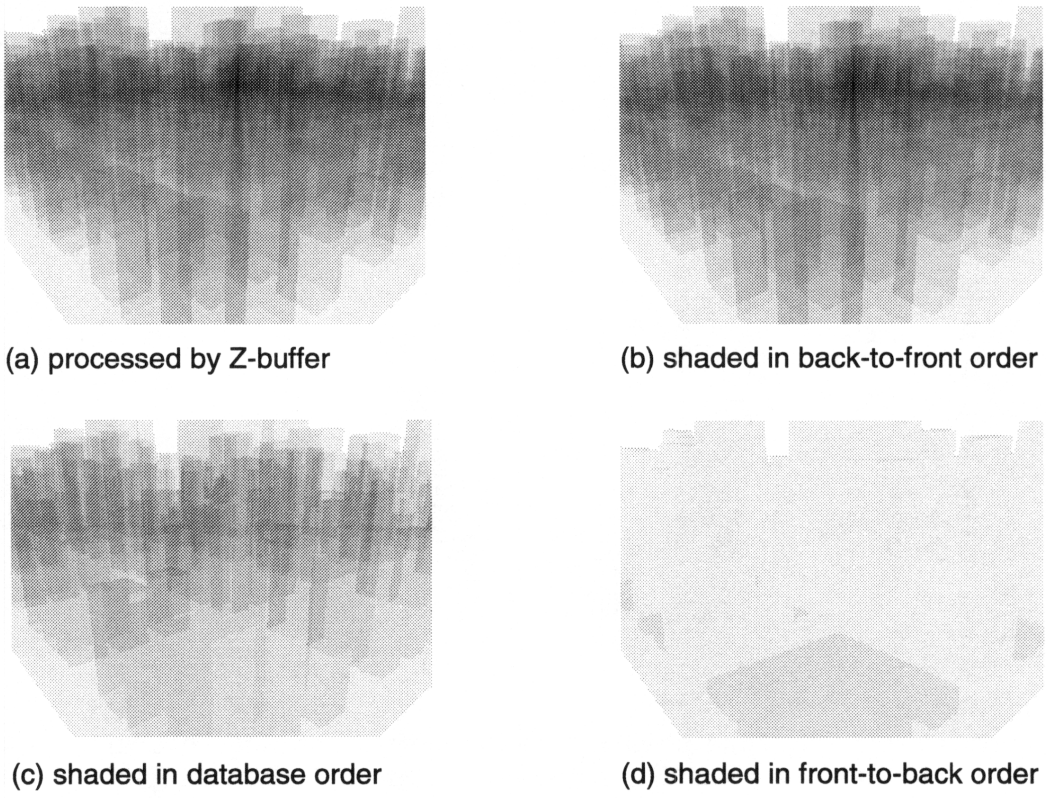


Figure 7.5: Summary of the observed performance of the Stereo Z-buffer algorithm with flat shading, smooth shading and texture mapping for each test scene. Error bars show the variation in performance for best (front-to-back) and worst (back-to-front) case depth orderings.

ing *and* shading type is considerable. For flat shading, the difference in rendering speed between the best (front-to-back) and worst (back-to-front) depth ordering is relatively small (but noticeable). For smooth shading, the variation is much greater, affecting rendering speed by up to about 50%. For texture mapping, it is greater still, with the overall rendering performance often varying by a factor of two or more.

To help understand why the depth ordering can have such a significant effect on the observed rendering performance with each shading method, it may be useful to consider the depth complexity of the scene. Figure 7.6(a) shows an overall depth complexity image for a single view taken from scene 1, representing the total work done by the visible surface algorithm at each point in the image. Figure 7.6(b) shows a shading depth complexity image for the same view for polygons processed in back-to-front depth order, representing the total work done by the shading process at each point in the image due to a pixel being considered visible at an intermediate step in the rendering process. Figure 7.6(c) shows a similar shading workload image for polygons processed in database order, while Figure 7.6(d) shows another



(a) processed by Z-buffer

(b) shaded in back-to-front order

(c) shaded in database order

(d) shaded in front-to-back order

Figure 7.6: Depth complexity images for a single view of scene 1 using the Stereo Z-buffer. Dark regions indicate relatively high depth complexity in the image; light regions indicate relatively low depth complexity.

shading workload image for polygons processed in front-to-back depth order. Note how much less work is done by the shading process for the front-to-back depth ordered image in (d) than either the database (c) or back-to-front (b) ordered images. Note also how similar the back-to-front ordered image is to the overall depth complexity image.

A summary of this effect for test scenes 1–5 is given in the chart of Figure 7.7. The visible depth complexities for each depth ordering as well as the overall depth complexity is shown for each scene. Given that a pixel is only shaded if it is considered visible at that point in the Z-buffer processing, it is clear that more shading is performed when more pixels are visible. When elements are processed in back-to-front order, more shading is performed than when using front-to-back depth order. Differences between the performance of the shading methods can readily be explained by comparing the computational complexity of each shading algorithm, with more costly shading techniques being subject to greater variation in performance with the amount of shading performed. By comparing Figure 7.7 with Fig-

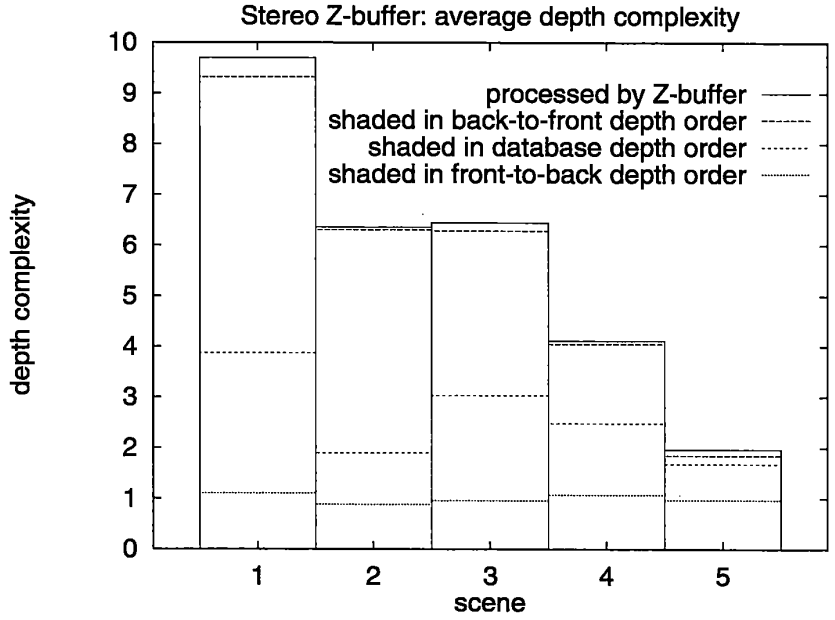


Figure 7.7: Average depth complexity levels for each test scene using the Stereo Z-buffer

ure 7.5, the close correspondence between shaded depth complexity and observed performance can be seen.

As a final point, it can be seen in Figure 7.7 how similar the visible depth complexity for the back-to-front depth ordering is to the overall depth complexity processed in each scene (the theoretical worst possible case for visible depth complexity). Note also how the visible depth complexity for the front-to-back depth ordering is close to an optimal value (for images which fill the viewing window) of 1.0 in all scenes. Together, these observations suggest that the front-to-back and back-to-front depth orderings used in these tests are good approximations of the theoretical best and worst case depth orderings for the Z-buffer algorithm.

7.1.2 Approximate stereo visibility

The current implementation of the Approximate Stereo Visibility algorithm does not have the same problem concerning the variation in rendering performance with depth ordering as the Stereo Z-buffer described in Section 7.1.1. This is due to the fact that it always processes primitives in front-to-back depth order, in order to obtain a reasonably reliable visibility estimate for the conditional reprojection of each

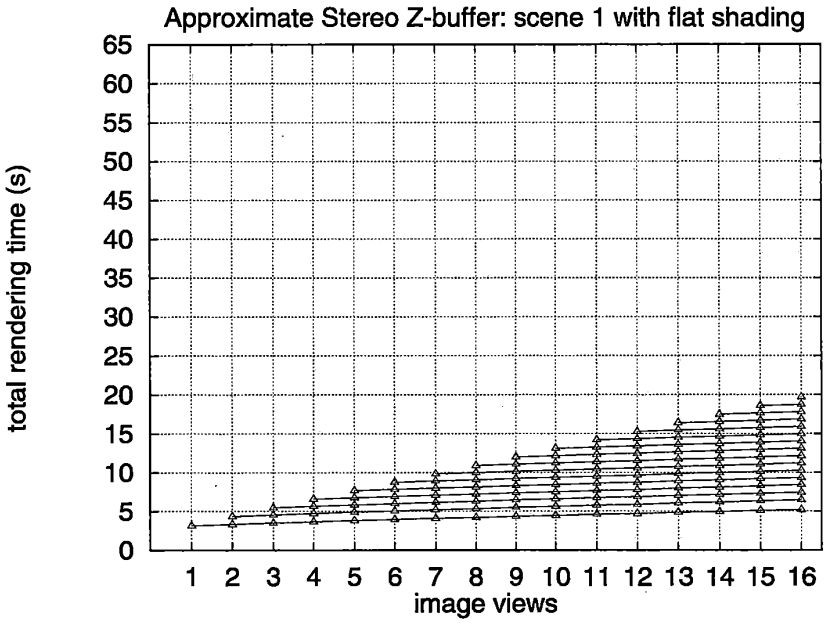


Figure 7.8: Observed performance of the Approximate Stereo Visibility algorithm for scene 1 with flat shading. The lowest trace represents the timings for a single sample view, the trace above that two sample views, and so on.

primitive in the approximate views.

In addition to varying the number of views to be rendered in the stereo image, it is possible to vary the number of views used to sample the visibility of each primitive. In the current implementation, the method used to allocate a given number of sample views from a set of image views is to distribute the sample views as evenly as possible among the image views, subject to the constraint that the outermost left and right views should be selected as sample views whenever possible (that is, for two or more sample views). This is intended to maximise the field-of-view covered by the sample views, and thus minimise the possibility of missing elements that should be visible. When only a single sample view is used, the view closest to the middle is chosen.

The observed performance of the approximate stereo visibility algorithm for scene 1 is shown in Figures 7.8, 7.9, and 7.10 for flat shading, smooth shading, and texture mapping respectively. In each graph the number of views rendered in the stereo image is plotted against the total rendering time recorded. Sixteen separate traces are shown in each graph, the lowest trace corresponding to the timings for one sample view, the trace above that for two sample views, then three sample

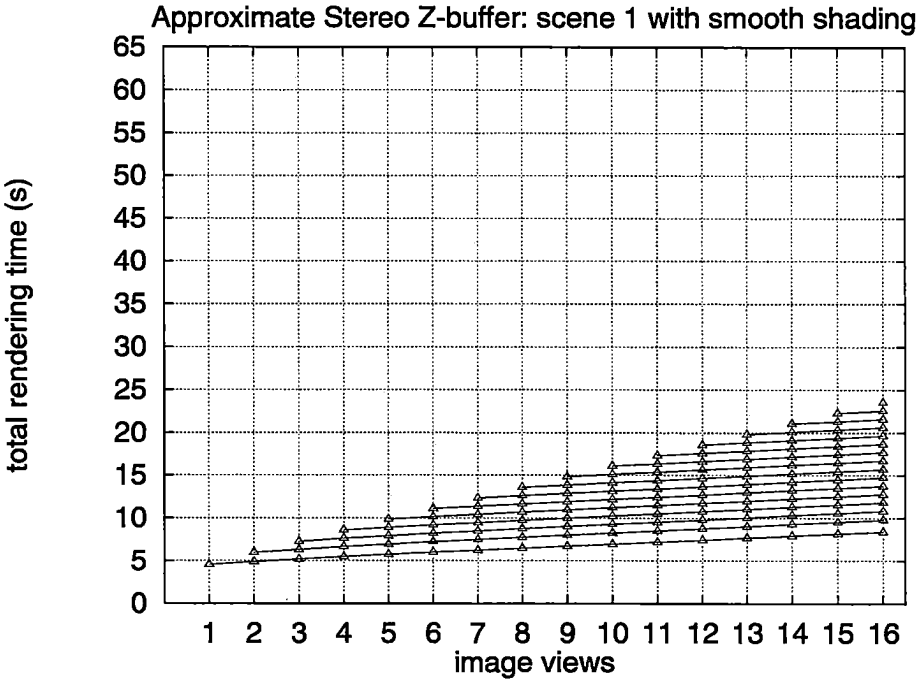


Figure 7.9: Observed performance of the Approximate Stereo Visibility algorithm for scene 1 with smooth shading. The lowest trace represents the timings for a single sample view, the trace above that two sample views, and so on.

views and so on. The graphs in Figures 7.8 to 7.10 are shown with the same vertical scale used in Figures 7.2 to 7.4 to facilitate comparisons between the performance of the Stereo Z-buffer and the Approximate Stereo Visibility algorithms.

From the data shown in the graphs of Figures 7.8 to 7.10 it can be seen that the relationship between the number of image views and total rendering time for a given number of sample views appears to be linear. It can also be seen that the total rendering time also depends on the number of sample views used as well as the shading method employed. However, in each graph the rendering time per approximated image view (slope of each trace) appears to be largely the same regardless of the number of sample views.

Similar observations may be made about the performance of the Approximate Stereo Visibility algorithm for scenes 2 to 5. It is possible to characterise this behaviour in terms of a simple linear model such as that in Equation 7.1, where t_v and t_o may depend on the number of sample views, shading method, scene and database used. If we then apply linear regression to the rendering performance data for each scene, shading method, and number of sample views, it is possible to obtain estimates of the rendering cost per approximated image view. The results of

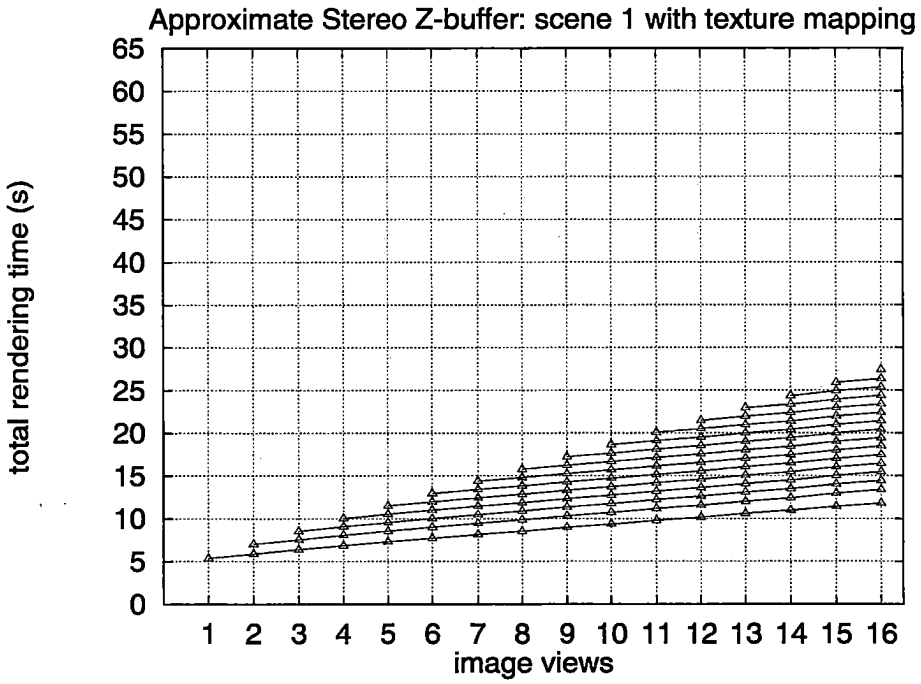


Figure 7.10: Observed performance of the Approximate Stereo Visibility algorithm for scene 1 with texture mapping. The lowest trace represents the timings for a single sample view, the trace above that two sample views, and so on.

this analysis are presented in Figure 7.11. The average rendering time per approximated image view is shown for each scene and shading method, with error bars representing the maximum observed variation in performance for different numbers of sample views. The vertical scale used in Figure 7.11 is the same as that used in Figure 7.5 to aid comparison.

It can be seen from Figure 7.11 that there is a significant difference in rendering speed between each of the shading methods. This difference is readily understood by comparing the relative computational cost of each shading method. In contrast, the variation of rendering performance between the different scenes is relatively small. Consideration of the rendering depth complexity for each scene may help understand this effect.

Figure 7.12 shows depth complexity images for sample and approximate views of scene 1. Figure 7.12(a) illustrates the visible surface processing workload at each point in one of the sample views, while Figure 7.12(b) shows the same workload for one of the approximated views. The work done in the approximated view is considerably less than that done in the sample view, as only elements which have been shown to be visible in one of the sample views are considered by the visible

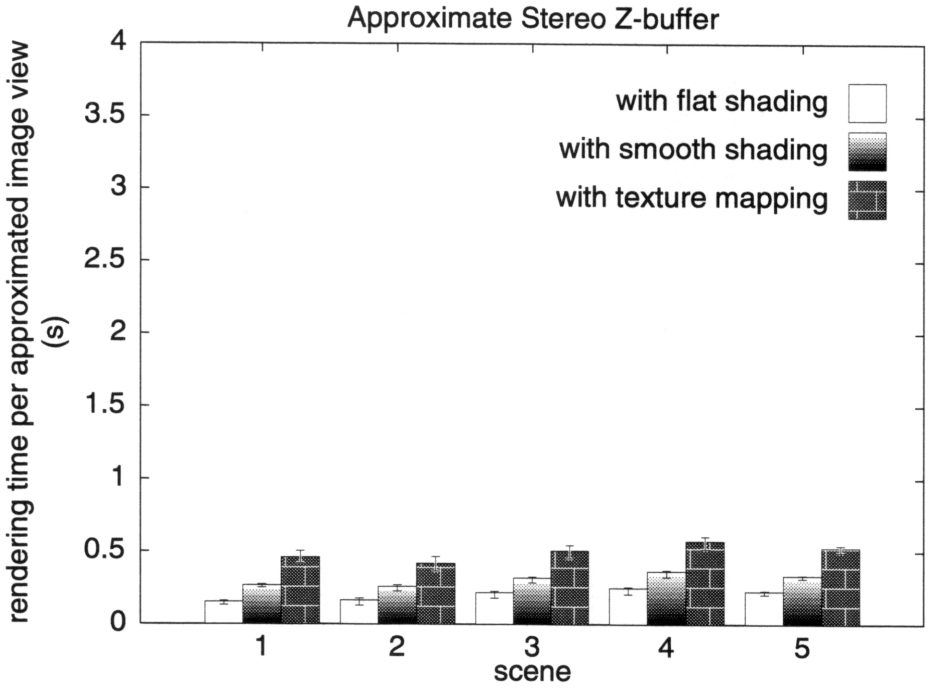
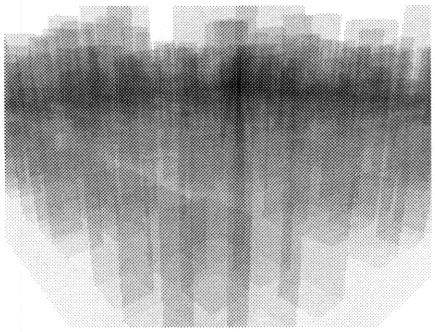


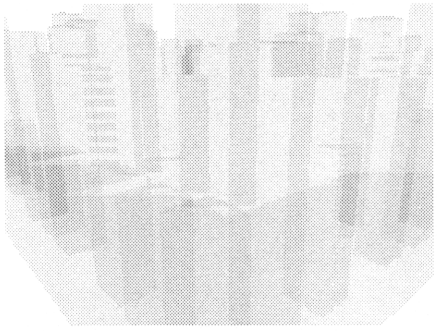
Figure 7.11: Summary of the observed performance of the Approximate Stereo Visibility algorithm with flat shading, smooth shading and texture mapping for each test scene. Error bars show the maximum variation in average rendering time per approximated view observed with different numbers of sample views.

surface process for the approximated view. On the other hand, the shading workload for the approximated view in Figure 7.12(d) appears very similar to that of the sample view in Figure 7.12(c). This may be explained by the fact that elements are processed in the same front-to-back depth order in both the sample and approximate views, and that the underlying principle of the Approximate Stereo Visibility algorithm is that what is considered visible in a sample view is likely to be visible in an approximate view as well.

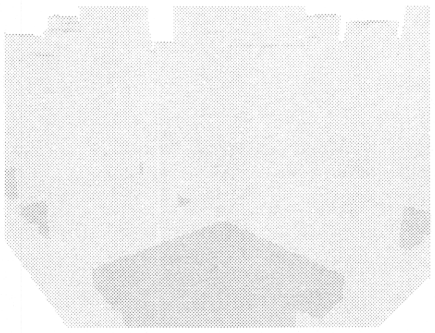
Figure 7.13 charts the processed and shaded depth complexities for sample and approximated views from each scene. This shows that while the actual depth complexity processed by the Z-buffer in each sample view varies considerably for each scene, the depth complexity processed in each approximate view is much the same for all scenes. Furthermore, both the depth complexity processed in the approximate views and the shaded depth complexity are far lower than the depth complexity processed in the sample views. This suggests that the Approximate Stereo Visibility algorithm provides an effective means of reducing the overall depth complexity that is processed to uniformly low levels, regardless of the depth complexity



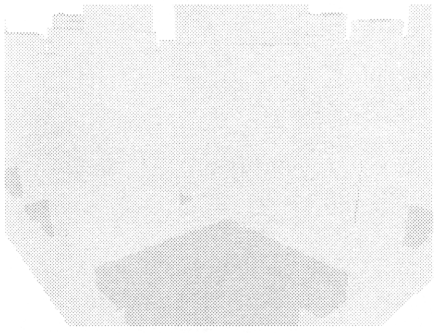
(a) processed by Z-buffer in sample view



(b) processed by Z-buffer in approximate view



(c) shaded in sample view



(d) shaded in approximate view

Figure 7.12: Depth complexity images for sample and approximate views from of scene 1 using Approximate Stereo Visibility. Dark regions indicate relatively high depth complexity in the image; light regions indicate relatively low depth complexity.

of the scene.

Apart from the question of rendering speed, the other major issue of concern with the Approximate Stereo Visibility algorithm is the quality of images produced by the technique. This was measured by comparing each approximate image with that obtained using the Stereo Z-buffer with the same scene, shading method and number of views. Figure 7.14 shows the results of such a pixel-by-pixel comparison for scene 3. The number of image views is plotted against the observed percentage of pixels which were found to differ between the approximate and accurate images. Each trace represents the observed differences for a different number of sample views, with the top trace using a single sample view, the next trace down using two sample views, and so on. The difference data is averaged for each of the three separate shading methods, with error bars at each data point demonstrating the range of differences detected by each method, although these variations are so

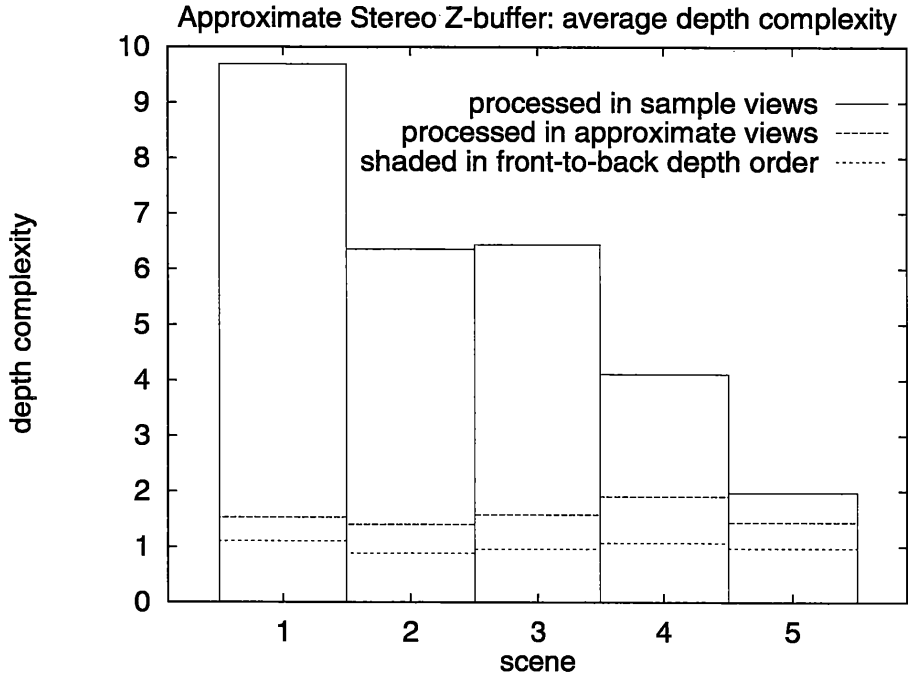


Figure 7.13: Average depth complexity levels for each test scene using Approximate Stereo Visibility

slight as to be all but unnoticeable in most cases.

From Figure 7.14, it appears that the fewer the number of sample views, the greater the measured differences. However, these differences tend to reach asymptotic levels as the number of image views increases. Similar observations about image differences may be made for the other scenes tested. The asymptotic difference level for each combination of scene and number of sample views may be estimated by calculating the weighted average of the observed image differences multiplied by the number of image views. A summary of the asymptotic image difference estimates obtained using this weighted average metric is presented in Figure 7.15. The number of sample views used is plotted against the estimated asymptotic difference level, with each trace representing a different scene.

It can be seen from Figure 7.15 that the estimated asymptotic image difference level falls off rapidly with increasing numbers of sample views. Indeed, for the test scenes used, only three or four sample views are required for the weighted average difference level to fall to less than 0.1% of the pixels in the image. With the test scenes used, increasing the number of sample views above four serves mainly to increase overall rendering cost for very little improvement in image quality. In this instance, the law of diminishing returns therefore suggests that using three or

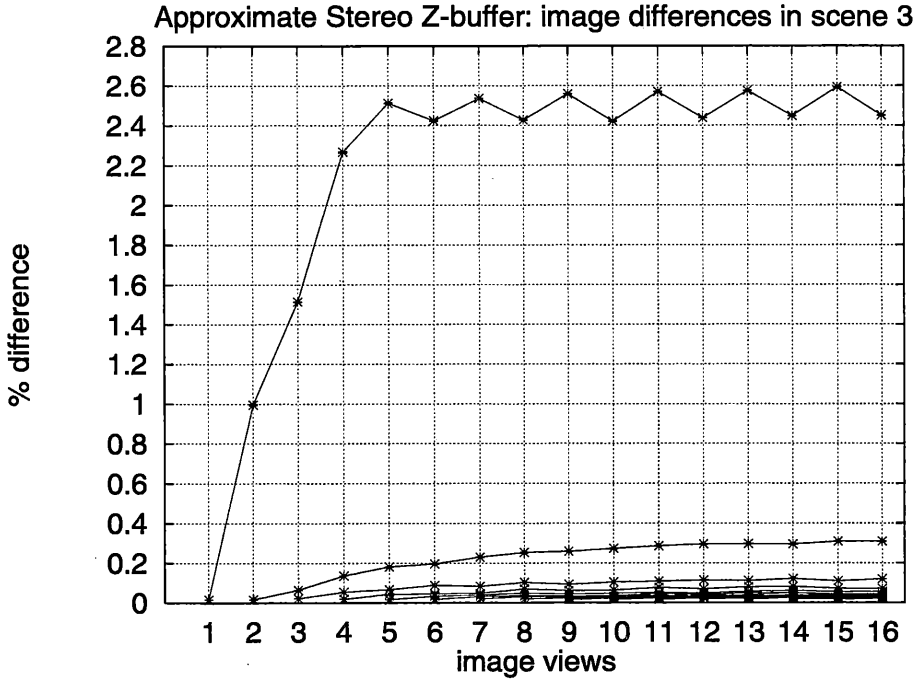


Figure 7.14: Observed differences between the approximate and accurate images of scene 3. The topmost trace represents the differences detected for approximate images using only one sample view, the next trace down the differences when using two sample views, and so on.

four sample views may provide the best compromise between rendering speed and quality.

7.2 Subjective image quality

The subjective image quality ratings given in this section are intended merely to provide a brief indication of how the images produced by each of the stereo rendering algorithms appear to an observer looking at the autostereo display and how they compare with each other. It presents only a personal opinion and does not represent the results of an empirical study into subjective image quality. First the use of multi-pass antialiasing for images produced by the Stereo Z-buffer algorithm is discussed, followed by some examples of images generated using the stereo viewpoint sampling technique described in Section 5.12.2. Finally the artifacts which appear in images rendered using Approximate Stereo Visibility are described, with reference to the observed image difference metrics presented in Section 7.1.2.

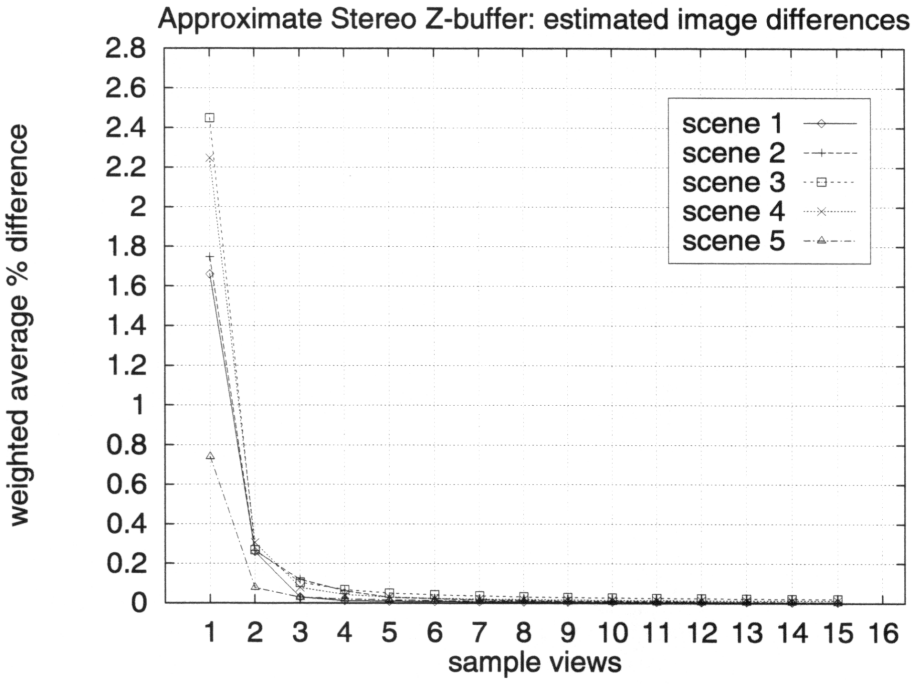


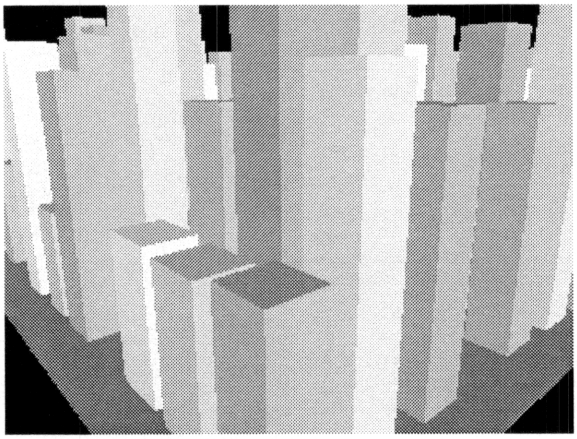
Figure 7.15: Estimated asymptotic image differences for a given number of sample views produced by Approximate Stereo Visibility. Each trace represents a different test scene.

7.2.1 Stereo Z-buffer image quality

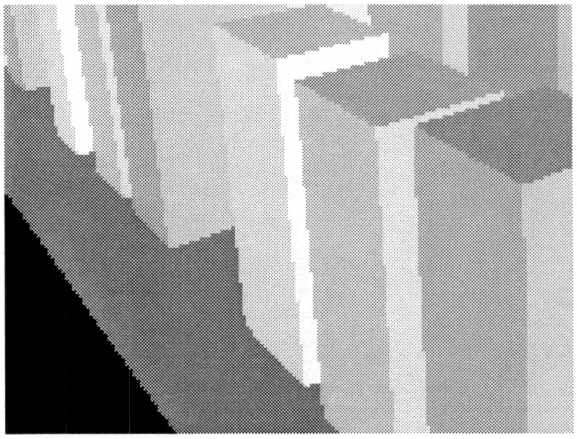
Point-sampled images rendered using the Stereo Z-buffer algorithm suffer from all the aliasing artifacts described in Section 3.3.3. To reduce the magnitude of these artifacts, antialiasing is applied using a multi-pass technique as described in Section 5.12.1.

The effect of varying the number of antialiasing passes in a flat shaded image is illustrated in Figure 7.16. The images in the column on the right show magnified details of the images in the column on the left. It can be seen that the staircasing artifacts visible in Figure 7.16(a) with only a single sample per pixel are considerably less noticeable after four passes in Figure 7.16(b). Little further improvement in image quality can be seen after eight passes, as in Figure 7.16(c).

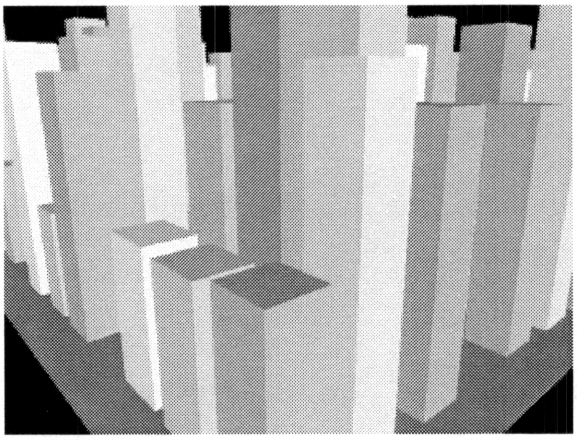
The effect of varying the number of antialiasing passes in a texture mapped image is illustrated in Figure 7.17. Severe aliasing artifacts such as dropouts and moiré patterns are present in Figure 7.17(a) after the first rendering pass. Although aliasing artifacts remain clearly visible in Figure 7.17(b), after four passes the overall noise level is considerably reduced. Subjective image quality continues to improve after eight passes (Figure 7.17(c)), with a discernable difference even after sixteen



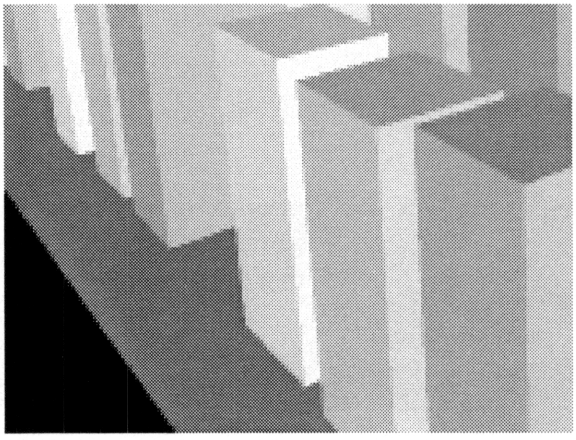
(a) one sample per pixel



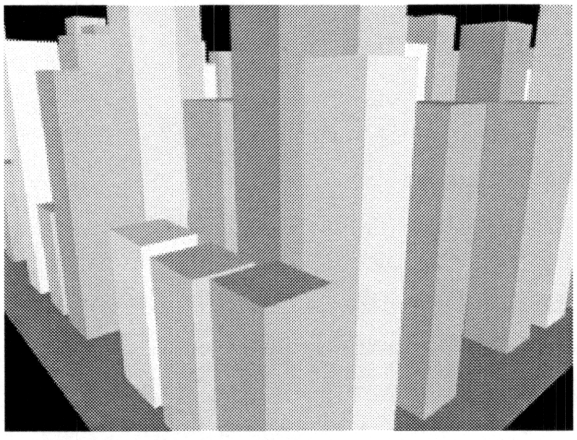
detail of (a)



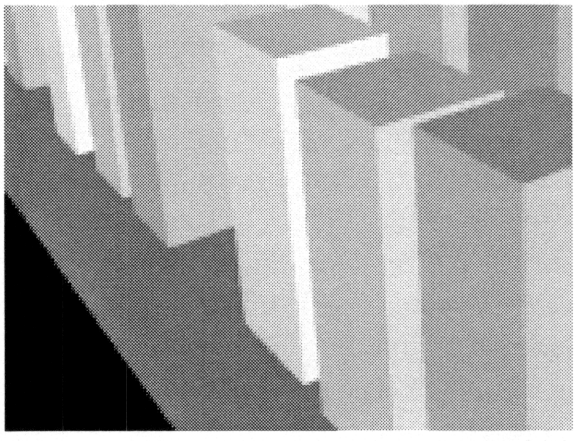
(b) four samples per pixel



detail of (b)



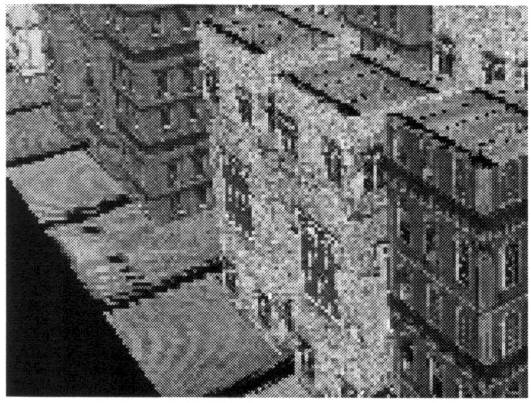
(c) eight samples per pixel



detail of (c)

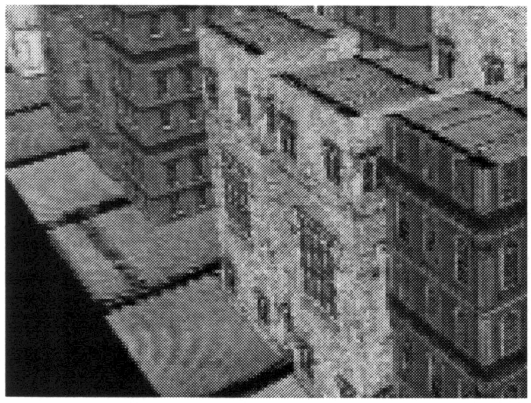
Figure 7.16: Multi-pass antialiasing with flat shaded polygons

(a) one sample per pixel



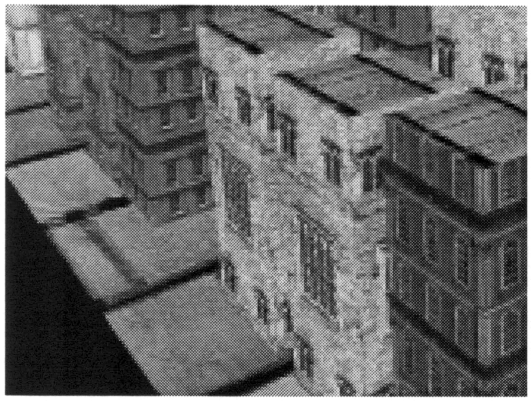
detail of (a)

(b) four samples per pixel



detail of (b)

(c) eight samples per pixel



detail of (c)

Figure 7.17: Multi-pass antialiasing with texture mapped polygons

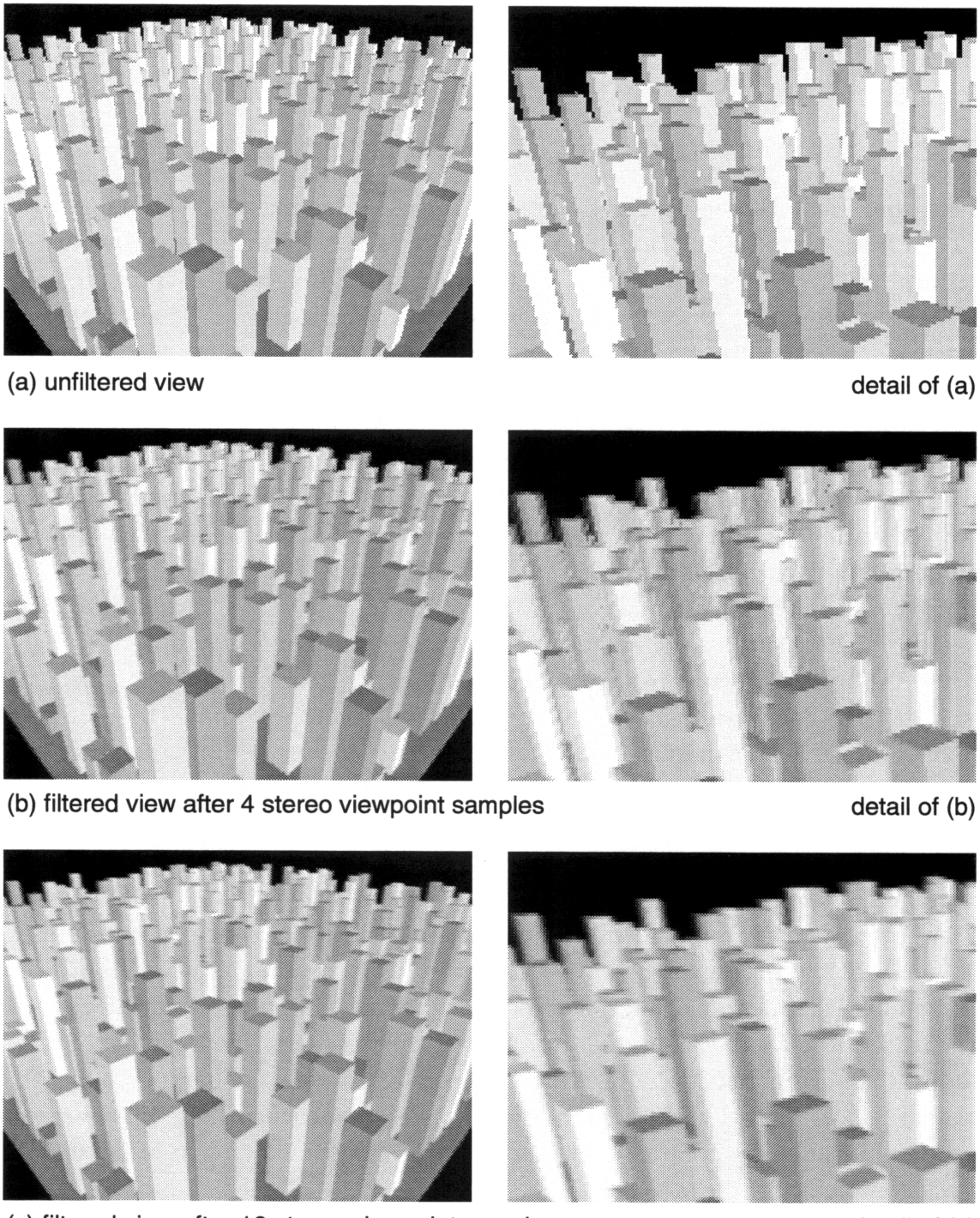
passes (not shown).

By comparing Figures 7.16 and 7.17, it can be seen that texture mapping requires more antialiasing passes than flat shading to obtain an image of subjectively similar quality. This may be attributed to the lack of filtering applied before sampling the texture map in the current implementation. Without such filtering, the number of passes that may be required to produce an image of acceptable quality may vary greatly depending on the scale of the texture map as it appears in the image. Even relatively crude texture filtering techniques may be suitable for use with a multi-pass approach, as any remaining aliasing artifacts will be subject to further filtering with each image accumulation pass. This was suggested by Haeberli and Akeley [1990], who proposed the use of standard MIP-mapping (Williams [1983]) but without the multi-sample interpolation between neighbouring levels of detail in the prefiltered texture map image, thus significantly reducing the computational cost of obtaining each textured pixel sample.

Using only a single viewpoint to generate each view in a stereo image results in a visibly "choppy" effect as the observer moves his head from side to side across the viewing zones in front of the display, owing to the discrete difference between each view. The effect is particularly noticeable for parts of the scene which appear a large distance from the view plane and thus have relatively large values of stereo parallax. Stereo viewpoint sampling (as described in Section 5.12.2) attempts to alleviate this by distributing the sample viewpoint location to different points within each viewing zone on separate rendering passes of the multi-pass antialiasing algorithm. Examples of the use of stereo viewpoint sampling are shown in Figure 7.18 for various numbers of antialiasing passes.

The subjective effect of this technique on an image is similar to conventional depth-of-field, except that the magnitude of the blurring is proportional to the depth of an object with respect to the viewing plane, rather than an arbitrary depth plane. Those parts of the image which are blurred the most are also those which exhibit the greatest stereo disparity between the views. This has the side effect of drawing the viewer's attention away from any elements which may be difficult to fuse stereoscopically owing to excessive stereo disparity. In addition, it also helps to smooth the transitions from one view and the next when the observer moves his head from side to side, "looking around" the image.

It can be seen by comparing Figure 7.18 with Figure 7.16 that many more passes may be required for stereo viewpoint sampling than for simple spatial antialiasing, in order to overcome the appearance of discrete "double images" rather than the desired continuous blurring. This is similar to the situation where multi-sample antialiasing is used for depth-of-field or motion blur effects, as described by Cook [1986] and Haeberli and Akeley [1990]. Cook [1986] suggests that stochastic sampling techniques applied at the level of each pixel independently can produce a



(a) unfiltered view

detail of (a)

(b) filtered view after 4 stereo viewpoint samples

detail of (b)

(c) filtered view after 16 stereo viewpoint samples

detail of (c)

Figure 7.18: Stereo viewpoint antialiasing using a multi-pass technique

higher quality antialiased image for a given number of samples per pixel. In contrast the approach used in multi-pass antialiasing only applies stochastic sampling at a much higher level, which makes it considerably more straightforward to implement but also correspondingly less efficient at dealing with complex phenomena.

7.2.2 Approximate Stereo Visibility image quality

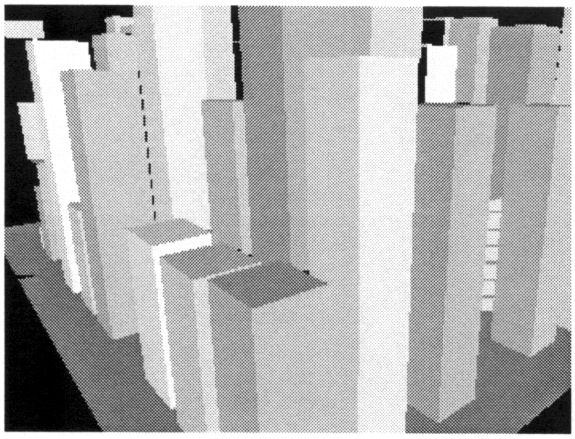
The Approximate Stereo Visibility algorithm produces an image which is made up of both fully sampled and approximated views. The fully sampled views are of identical quality to those produced by the Stereo Z-buffer, but the approximated views may exhibit a number of artifacts. These artifacts may all be attributed to certain elements of the scene being omitted by the algorithm because they were not visible in any of the sample views.

An example which demonstrates these artifacts for an approximated view of scene 1 are shown in Figure 7.19. The view in Figure 7.19(a) is the one produced by Approximate Stereo Visibility, with the correct view in (b) obtained using the Stereo Z-buffer. Figure 7.19(c) shows where the differences between (a) and (b) are found in the view, while (d) is the sample view used to estimate visibility for the approximate view in (a).

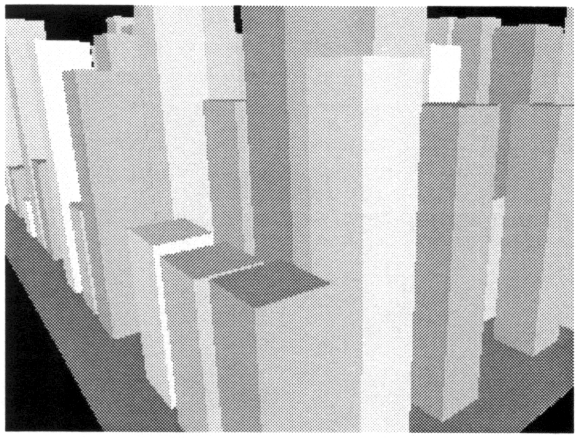
Figure 7.20 illustrates how increasing the number of sample views used can help reduce the apparent errors in approximated views. The correct view (obtained using the Stereo Z-buffer) is shown in Figure 7.20(a). An approximated view from a 16-view image is shown in the left column, with its corresponding difference image on the right. The number of sample views used increases down the page, with one sample view in Figure 7.20(b), two sample views in Figure 7.20(c), and three sample views in Figure 7.20(d). Note how rapidly the differences are reduced as the number of sample views increases, and how similar the approximated view is to the correct one after only three sample views. This supports the observed image difference data presented in Section 7.1.2 and summarised in Figure 7.15.

Although it appears clear that the frequency of errors in the approximated views may be reduced by increasing the number of sample views, the question of how readily such errors may be noticed by an observer may also depend on a variety of other factors related to the content of the scene. In particular these may include the colour or intensity contrast of neighbouring regions in the image, and the local visual complexity of the scene.

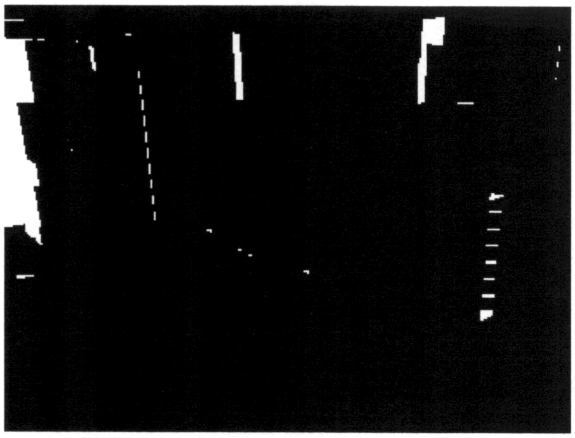
For example, consider a missing element in an approximated view which reveals an object which should not be visible in that view. If the colour of the revealed object contrasts with those around it, it is much more likely to be noticed than if it blends in better with its surroundings. On the other hand, errors affecting only a few sparsely-distributed pixels in an image with high textural or geometric com-



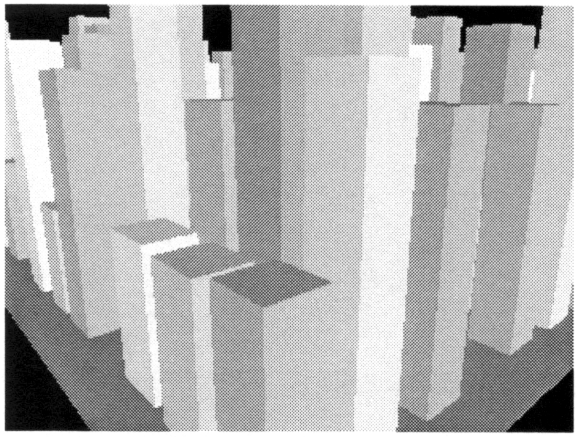
(a) approximated view



(b) correct view



(c) difference between approximate and correct views



(d) sample view used to estimate visibility

Figure 7.19: Artifacts due to missing scene elements in an approximate view of scene 1

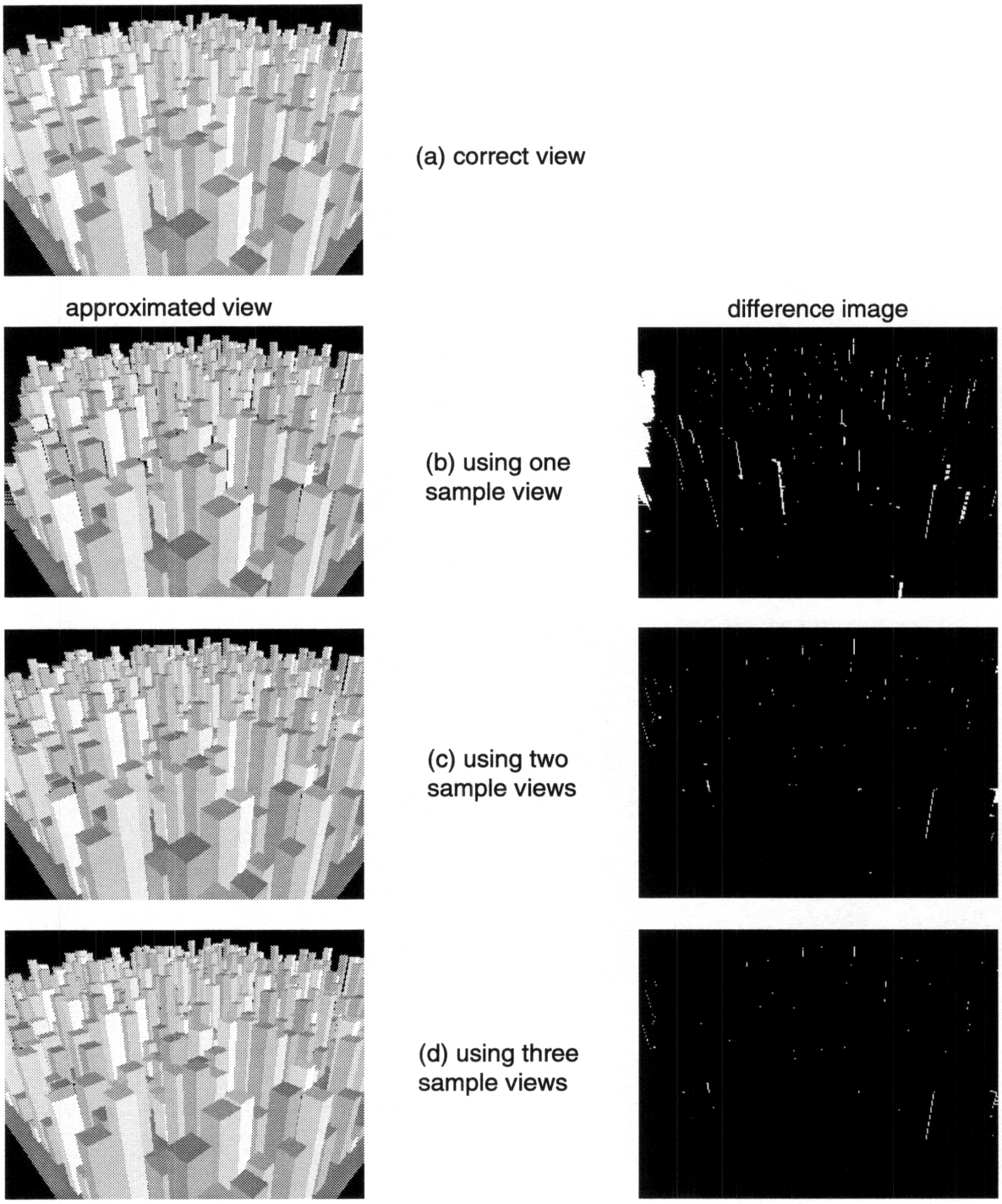


Figure 7.20: Reducing artifacts in approximate views by increasing the number of sample views used to estimate visibility of elements in the scene

plexity are more likely to be tolerated than in a scene consisting of a few simple flat shaded geometric shapes.

Although only point-sampled images have been discussed so far in this section, the Approximate Stereo Visibility algorithm is capable of supporting multi-pass antialiasing in the same way as the Stereo Z-buffer described in Section 7.2.1. Of course, it may be questionable whether it is sensible to even consider antialiasing approximated images. Nonetheless, if the difference between the approximated and fully sampled images is sufficiently small, then the benefits of the increased rendering speed with the approximate algorithm may well outweigh the slight loss of image quality and provide a useful performance boost for the multi-pass antialiasing approach.

7.3 Discussion

In this section, the results presented in Sections 7.1 and 7.2 are reviewed and their implications discussed, together with suggestions for further research in the area.

7.3.1 Stereo Z-buffer

The observed performance results for the Stereo Z-buffer algorithm presented in Section 7.1.1 demonstrate a link between the depth complexity of the scene (Figure 7.7) and the rendering cost for each view in the stereo image (Figure 7.5). They also show how the rendering performance of the algorithm may vary significantly depending on the depth order in which primitives are processed, as well as with the shading method used.

The relationship between depth complexity and rendering performance is of course a well-known phenomenon, with depth complexity recognised by Sutherland et al. [1974] as a characteristic measure of the difficulty of visible surface determination for any given scene. The practical implication of this relationship is that it is possible to improve rendering performance by reducing the effective depth complexity. Processing the scene database in front-to-back depth order capitalises on this effect by decreasing the proportion of pixels that are considered visible by the Z-buffer, thus eliminating many unnecessary shading computations. The more expensive the shading technique used and the greater the overall depth complexity of the scene, the more the front-to-back processing strategy pays off. In the experiments described here, the observed rendering performance for the best (front-to-back) and worst (back-to-front) depth orderings typically varied by a factor of two or more for texture mapping, yet only by a few percent for flat shading. With more complex scenes or more sophisticated shading methods, greater differences would

be expected.

The dependence on depth ordering may present significant problems when attempting to analyse the performance of the algorithm and compare it with others. Indeed, the magnitude of the variation in observed performance with depth ordering is so great for the more expensive shading methods used here that it can be difficult to make meaningful comparisons with alternative rendering algorithms, unless this effect is taken into account. An equivalent implementation of a similar Z-buffer visible surface algorithm which computes shading information for each pixel irrespective of visibility would not exhibit this same variability, but for an entirely software-based renderer this would mean sacrificing performance to a level corresponding to the worst case observed in these experiments. If the penalty incurred by unconditional shading is considered unacceptable and performance is the most important consideration, then conditional shading should be used and primitives processed in front-to-back depth order.

Although this ordering was obtained using an explicit depth sorting step prior to actual processing by the Z-buffer in the current implementation, it may be possible to avoid this by structuring the scene database in a suitable spatial representation which permits traversing the scene in the desired order[1]. Similar ideas involving combinations of object space and image space techniques have been explored by Gordon and Chen [1991] and Greene et al. [1993]. Gordon and Chen [1991] used front-to-back traversal of a BSP tree with a "dynamic screen" masking technique to prevent writing to any given pixel more than once. Greene et al. [1993] described a Z-buffer variant that uses an object space octree for the database representation in conjunction with an image space Z-pyramid to quickly reject large hidden portions of the scene, thus significantly reducing rendering times for highly complex databases. While the object space components of both these techniques would benefit from having their costs shared among all the views in a multi-view stereo image, the actual image space visible surface determination would still have to be done separately for each view.

7.3.2 Approximate Stereo Visibility

For the Approximate Stereo Visibility algorithm, the performance results given in Section 7.1.2 suggest a correlation between the depth complexity of the scene processed in the approximated views of the stereo image (Figure 7.13) and the rendering cost for each approximated view (Figure 7.11). The experimental data also indicates that the relative frequency of errors detected in the approximated image de-

[1] the front-to-back depth ordering does not have to be particularly precise to be useful, as illustrated by the results obtained with the approximate bucket sorting technique used in these experiments.

creases rapidly to asymptotic low levels with increasing numbers of sample views (Figures 7.14 and 7.15).

While the relationship between depth complexity and rendering cost in the approximated views is readily understood in terms similar to those already explored in Section 7.3.1, the critical accompanying observation is that the rendering cost in the approximated views shown in Figure 7.11 does not appear to be related to the overall depth complexity of the scene as a whole, as represented in Figure 7.13 by the depth complexity processed in the sample views. The significance of this result is that the performance of Approximate Stereo Visibility in the approximated views is largely independent of the depth complexity of the scene, and thus the relative speedups possible with this technique increase in proportion to the depth complexity of the scene. By using negative visibility results from the sample views to prevent portions of the original scene database from even being considered in the approximate views, this strategy effectively provides a means of reducing the overall average depth complexity processed by the renderer. The greater the ratio of approximate views to sample views, the closer the overall average depth complexity will become to the depth complexity processed in the approximate views, and thus the closer the average rendering performance per view will become to the performance in the approximate views.

Although it uses the same Z-buffer visible surface determination method as the Stereo Z-buffer, the Approximate Stereo Visibility algorithm does not suffer from similar variations in performance owing to relative depth ordering because it routinely processes primitives from front-to-back so as to obtain a reasonably reliable estimate of visibility of each element. Here too an explicit depth sorting step may be avoided by using a suitable spatial representation of the scene database to allow its traversal in the desired front-to-back order, as mentioned in Section 7.3.1.

Errors in any given approximate view are due to elements in the scene which are not visible in any of the sample views, but which should appear in that approximate view. These missing elements may be thought of as falling into "blind spots" which are hidden by other parts of the scene in all the sample views[2]. Increasing the number of sample views thus decreases the extent of the blind spots, which in turn reduces the probability of errors due to missing objects in the approximate views. However, it is important to realise that it is not possible (in general) to entirely eliminate errors in the approximate views using this strategy, as a suitably complex database can always be constructed which exhibits the appropriate artifact[3] for any given set of sample viewpoints.

[2]Chen and Williams [1993] describe a similar problem of "holes" in an interpolated image due to objects which are located in the umbra (full shadow) region that would be cast by the set of sample viewpoints if they were point light sources.

[3]such as the narrow concave region shown in Figure 6.5.

The trade off between the total rendering time and the approximated image error rate appears to reach optimum levels with three or four sample views for the test scenes used. The actual optimum number of sample views needed by the Approximate Stereo Visibility algorithm may vary, depending not only on the content of the scene but also on the stereo field-of-view between the leftmost and rightmost views — a fixed value of 15° in the experiments described here, corresponding to the stereo field-of-view of the current version of the autostereo display. Indeed, it is hypothesised that the critical parameter is not the number of sample views *per se*, but rather the angular density of the distribution of those sample views over the stereo field-of-view of the image. If this hypothesis is correct, it would suggest that an arbitrary number of intermediate views could be synthesised from a relatively small set of sample views, where the cost and quality of each intermediate view could reasonably be expected to be similar to those obtained in the experiments described here. As the number of views in the image increases, the performance benefits of using approximations tends to outweigh the penalty in terms of image quality, and such an approach becomes increasingly attractive. In a separate but related issue, it remains to be seen whether the use of a wider stereo field-of-view (which may accompany any increases in the number of views in the image) has any effect on the way in which visibility information is shared between the views[4].

7.3.3 Comparison of the two techniques

By comparing Figure 7.5 with Figure 7.11, it can be seen that the Approximate Stereo Visibility algorithm generally has a lower rendering cost per image view than the Stereo Z-buffer. However, the magnitude of the speedup it offers varies considerably, depending on the depth complexity of the scene: the greater the depth complexity, the greater the performance advantage enjoyed by Approximate Stereo Visibility. Because Approximate Stereo Visibility is designed to eliminate as many of the hidden surfaces as possible from consideration by the visible surface determination algorithm in the approximated views, its performance in the approximate views varies only relatively slightly with different scene depth complexities in comparison with the Stereo Z-buffer. The success of Approximate Stereo Visibility in achieving this is illustrated in Figure 7.13, which shows that the depth complexity processed in the approximate views is not only generally much less than that processed in the sample views, but also exhibits much less variability.

As the number of views in the image increases, the shared processing overheads of the Stereo Z-buffer and Approximate Stereo Visibility become more sparsely distributed among the views. With a large enough number of views, the average ren-

[4]for example, it may be helpful to partition the stereo field-of-view to enable the use of more localised measures of visibility.

dering cost per view asymptotically approaches the performance levels summarised in Figures 7.5 and 7.11. In contrast, repeated application of conventional single-view rendering techniques for multi-view stereo incurs the same cost per view regardless of the number of views in the image, owing to the way in which each view is processed independently of all the others. To put this in context, Figure 7.21 shows the total time taken by the Stereo Z-buffer to render only a single-view image of each test scene with flat shading, smooth shading and texture mapping. While it is important to emphasize that these timings include the cost of performing stereo disparity calculations which are not needed for a single-view image[5], they may be taken as a rough guide to the performance of a similar conventional single-view Z-buffer algorithm. Comparison of Figure 7.21 with Figures 7.5 and 7.11 (all shown at the same vertical scale) gives an indication of the magnitude of the savings possible using stereo rendering techniques, but further work is needed to more accurately quantify the benefits.

7.3.4 Memory requirements

One issue which has not yet been considered here is the question of the memory requirements of these stereo rendering algorithms. Both algorithms rely on a Z-buffer for resolving visible surfaces independently for each view of the multi-view stereo image. In the tests described in this chapter, the relatively low resolution of each view (320×240 pixels) and of the Z-buffer used (one 16-bit Z-value per pixel) has allowed the amount of memory required to support this to be kept within manageable limits. However, this cost may become prohibitive for images of higher resolution or greater Z-buffer precision. For example, an 8-view 640×480 image with 32 bits per Z-value would require 9.8 Mbyte of Z-buffer storage, in comparison with the 2.4 Mbyte required for the 16-view 320×240 pixel 16-bit Z-buffer used in the experiments described in this chapter.

In order to address this issue, both the Stereo Z-buffer and the Approximate Stereo Visibility algorithms have also been tested using a scanline (rather than full screen) Z-buffer implementation. This has significantly reduced the memory required for the Z-buffer, to only 10 Kbyte for the 16-view 320-pixel 16-bit scanline Z-buffer used for the test scenes. Furthermore, the observed performance of the scanline-based Z-buffer algorithms is very similar to that of the full screen Z-buffer versions in all respects, including rendering time per view and error rate in the approximated images. This approach allows higher resolution images with greater

[5] through analysis of the current implementation of the Stereo Z-buffer algorithm, it is estimated that these stereo disparity calculations may account for no more than 25% of the processing required to scan convert each polygon in the worst possible case, and are typically expected to consume considerably less effort.

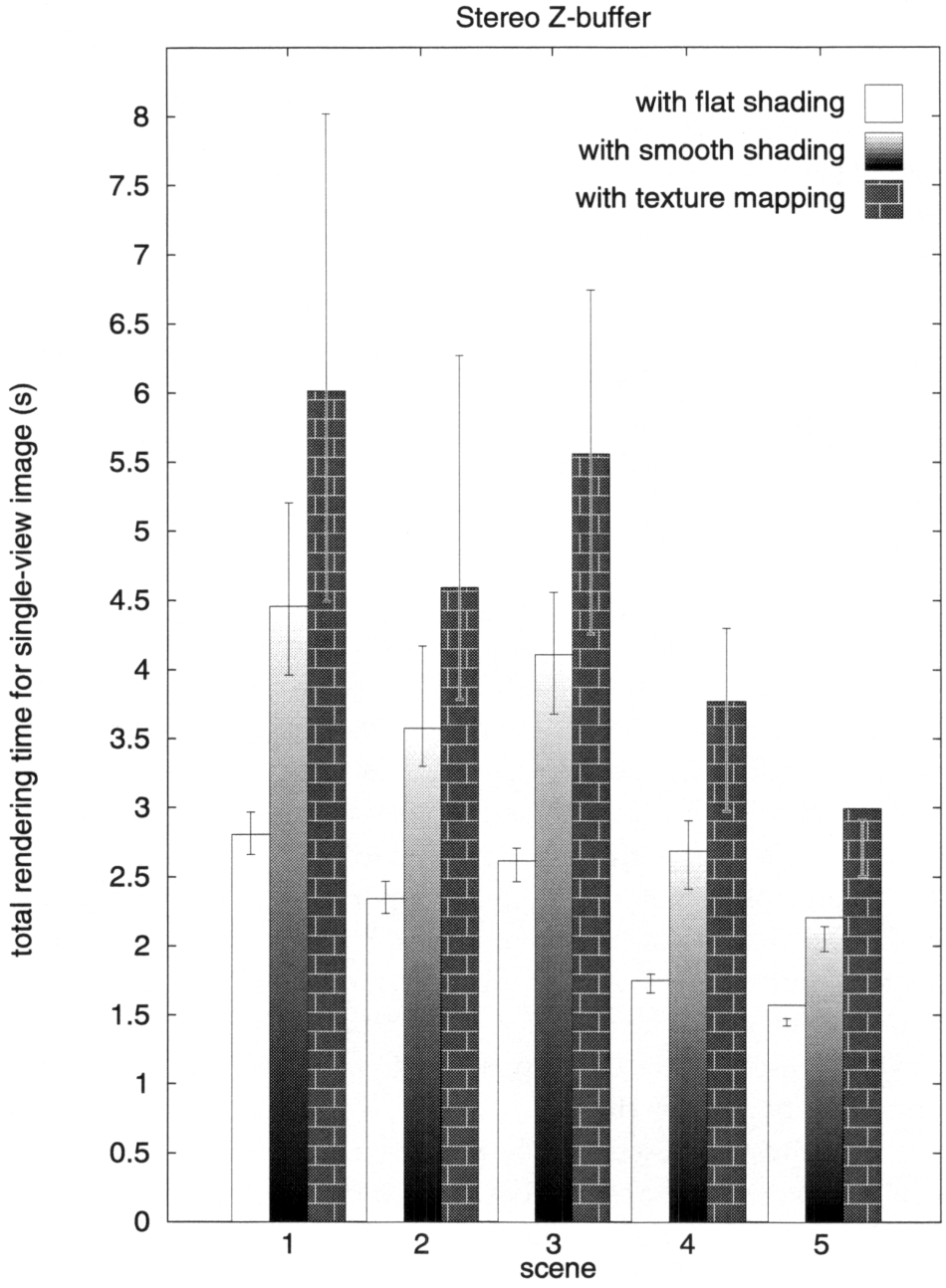


Figure 7.21: Observed performance of the Stereo Z-buffer for only a single-view image. Error bars show the maximum variation in total rendering time observed for best (front-to-back) and worst (back-to-front) case depth orderings. The apparently anomalous error bars in the timings for scene 5 show that processing the database in back-to-front depth order does not necessarily produce the worst performance in all cases, particularly where the database is not isotropic (that is, the database does not exhibit an even spatial distribution in all three dimensions).

Z-buffer precision to be accommodated at a relatively modest cost in terms of Z-buffer memory, although these advantages are offset to some extent by the storage required to sort all the polygons in the scene into the scanline order demanded by the scanline approach. It is expected that these considerations may become even more important as the number of views in the image increases. Indeed, such techniques could be readily adapted to generate source images for holographic stereograms which typically consist of around one hundred perspective views (Guo et al. [1988]; Halle [1994]).

7.3.5 Further work

Although the Approximate Stereo Visibility algorithm seems to offer a better prospect for future research into multi-view stereo rendering techniques than the Stereo Z-buffer, the errors it is susceptible to in the approximated views, although relatively small, are (at best) undesirable and may even make it unacceptable for some applications. Furthermore, at present there is no formal means of determining the level of these errors[6], and the only means of reducing them — increasing the number of sample views used in the image — is relatively inefficient and expensive for more than a few additional sample views[7].

It may be possible to adapt a variation on a scanline visible surface algorithm such as that of Crocker [1984] to work in stereo in a manner similar to that of the Approximate Stereo Visibility algorithm. In this scheme, it is envisaged that elements visible in one view might be tagged for priority consideration in the next as being likely to be visible there also, but are not discarded altogether if not visible in the first instance. Instead, all elements in the scene are retained for possible processing in all views, with the desired speedups obtained by partitioning the scene according to the probability of each element that is visible in one view being visible in the next. The adoption of such a strategy would have the advantage of producing an exact rather than approximate image, and would thus eliminate all concern of unwanted errors appearing in the image. In many ways however, this is equivalent to explicitly solving the visible surface problem for each view. An alternative approach is to use the stereo reprojection error correction technique described by Adelson and Hodges [1993], although it is not clear whether this method can be successfully adapted from its original ray tracing form to other types of renderer.

[6] except by computing the correct image by some other means and performing a pixel by pixel comparison, which only makes sense for test purposes.

[7] this may be an artifact of the relatively small number of views in the images used with the current autostereo display — with more views in the image, if the same number of sample views can be used to generate a greater number of approximated views, then this may become a relatively inexpensive way to reduce the error frequency.

8

Conclusions

This thesis has addressed some of the challenges faced by synthetic image generation for a novel multi-view autostereo display device. This chapter summarises the results of this research and the conclusions reached, as well as suggesting where further work may be beneficial.

8.1 Summary

The technological background behind this work was presented first, with a brief description of the Cambridge Autostereo Display and the experimental computer graphics platform developed to support it. Next, the general problems faced by synthetic image generation were reviewed, of which visible surface determination was identified as the most important for multi-view stereo. A multi-view autostereo viewing model was then derived, guided by existing conventional monoscopic and two-view stereoscopic viewing models.

Based on this autostereo viewing model, two new algorithms for multi-view stereo image synthesis were developed. The Stereo Z-buffer described in Chapter 5 is an extension of conventional single-view rendering techniques, adapted and optimised to take advantage of the coherence between the views in a multi-view stereo image. The Approximate Stereo Visibility algorithm described in Chapter 6 produces an approximate multi-view stereo image by sharing visibility information between the views, in an attempt to reduce visible surface processing in those parts

of the scene which are considered unlikely to have any noticeable affect on the appearance of the final image.

An experimental evaluation of these new techniques was presented in Chapter 7. This showed that the Stereo Z-buffer is capable of generating multi-view stereo images more efficiently than a single-view image on a cost per view basis, although considerable variation in observed performance was found when processing the database in different depth orderings. It also demonstrated that the Approximate Stereo Visibility algorithm is capable of producing approximate multi-view stereo images (of comparable quality) more efficiently than the Stereo Z-buffer.

8.2 Conclusions and further work

It has been shown that it is possible to synthesise multi-view stereo images more efficiently by taking account of the similarities between the views than by rendering each view independently. Two different approaches to this problem have been studied: the Stereo Z-buffer algorithm uses conventional single-view rendering techniques adapted and optimised for multi-view stereo images, while the Approximate Stereo Visibility algorithm trades improvements in rendering speed against the probability of errors in the final image. Experimental results indicate that the potential performance benefits available using these stereo rendering algorithms increases with the number of views in the image, reaching asymptotic levels as the shared processing costs become relatively insignificant compared with the overall rendering time. The depth order that elements in the scene database are processed in was also found to have a significant impact on the observed rendering performance, with front-to-back processing resulting in considerably less wasted shading effort for the Z-buffer visible surface algorithm.

In general terms, these results also show that the performance of both stereo rendering algorithms is related to the depth complexity of the scene being rendered. Thus the Approximate Stereo Visibility algorithm offers much greater potential performance improvements than the Stereo Z-buffer, as it actively helps reduce the depth complexity processed in the approximated views. The main problem with the approximate approach is the unwanted errors in the approximated views, which are currently both impractical to detect and relatively expensive to try to reduce.

As multi-view stereo display technology improves and the number of views required increases, the performance advantages of using techniques such as Approximate Stereo Visibility is expected to outweigh the cost in terms of image quality, particularly for interactive applications. If it is possible to generate a large number of closely-spaced approximate views of similar quality from a relatively small

number of sample views (as it has been hypothesised), then such approximate techniques may become even more attractive. It may be possible to develop reliable methods of detecting and correcting such errors in the image, although it may prove difficult in general to satisfy the image quality requirements without sacrificing too much performance.

Bibliography

Abram, G. and Westover, L. (1985). Efficient alias-free rendering using bit-masks and look-up tables. In *Computer Graphics: Proceedings of SIGGRAPH '85*, Vol. 19, No. 3, pages 53–59.
(cited on page 52)

Adelson, S. J., Bentley, J. B., Chong, I. S., Hodges, L. F., and Winograd, J. (1991). Simultaneous generation of stereoscopic views. *Computer Graphics Forum*, Vol. 10, No. 1, pages 3–10.
(cited on pages 47, 90, 108, 112)

Adelson, S. J. and Hodges, L. F. (1993). Stereoscopic ray-tracing. *The Visual Computer*, Vol. 10, No. 3, pages 127–144.
(cited on pages 47, 90, 126, 127, 129, 130, 167)

Akeley, K. (1993). RealityEngine graphics. In *Computer Graphics: Proceedings of SIGGRAPH '93*, Annual Conference Series, pages 109–116.
(cited on page 118)

Akka, R. (1992). Automatic software control of display parameters for stereoscopic graphics images. In *Stereoscopic Displays and Applications III*, Proc. SPIE 1669, pages 31–38.
(cited on page 77)

Asal, M., Short, G., Preston, T., Simpson, R., Roskell, D., and Guttag, K. (1986). The Texas Instruments 34010 Graphics System Processor. *IEEE Computer Graphics and Applications*, Vol. 6, No. 10, pages 24–39.
(cited on page 30)

Badt, Jr., S. (1988). Two algorithms for taking advantage of temporal coherence in ray tracing. *The Visual Computer*, Vol. 4, No. 3, pages 123–132.
(cited on page 124)

Baker, J. (1987). Generating images for a time-multiplexed stereoscopic computer graphics system. In *True 3D Imaging Techniques and Display Technologies*, Proc. SPIE 761, pages 44–52.
(cited on pages 64, 66, 69)

Barkans, A. C. (1991). Hardware-assisted polygon antialiasing. *IEEE Computer Graphics and Applications*, Vol. 11, No. 1, pages 80–88.
(cited on pages 55, 118)

Benton, S. A. (1982). Survey of holographic stereograms. In *Processing and Display of Three-Dimensional Data*, Proc. SPIE 367, pages 15–19.
(cited on page 78)

Blinn, J. F. and Newell, M. E. (1976). Texture and reflection in computer generated images. *Communications of the ACM*, Vol. 19, No. 10, pages 542–547.
(cited on page 45)

Brewster, D. (1856). *The Stereoscope: its history, theory and construction*. John Murray, London.
(cited on page 4)

Butts, D. R. W. and McAllister, D. F. (1988). Implementation of true 3D cursors in computer graphics. In *Three-Dimensional Imaging and Remote Sensing Imaging*, Proc. SPIE 902, pages 74–84.
(cited on pages 14, 66)

Carpenter, L. (1984). The A-buffer, an antialiased hidden surface method. In *Computer Graphics: Proceedings of SIGGRAPH '84*, Vol. 18, No. 3, pages 103–108.
(cited on pages 44, 52)

Catmull, E. (1974). *A Subdivision Algorithm for Computer Display of Curved Surfaces*. PhD thesis, Computer Science Department, University of Utah, Salt Lake City, UT. published as Technical Report UTEC-CSC-74-133.
(cited on pages 42, 44)

Chapman, J., Calvert, T. W., and Dill, J. (1991). Spatio-temporal coherence in ray tracing. In *Proceedings of Graphics Interface '91*, pages 101–108.
(cited on page 46)

Chen, S. E. and Williams, L. (1993). View interpolation for image synthesis. In *Computer Graphics: Proceedings of SIGGRAPH '93*, Annual Conference Series, pages 279–288.
(cited on pages 47, 90, 126, 127, 129, 130, 135, 163)

Cohen, M. F. and Greenberg, D. P. (1985). The hemi-cube: A radiosity solution for complex environments. In *Computer Graphics: Proceedings of SIGGRAPH '85*, Vol. 19, No. 3, pages 31–40.
(cited on page 96)

Cook, R. L. (1986). Stochastic sampling in computer graphics. *ACM Transactions on Graphics*, Vol. 5, No. 1, pages 51–72.
(cited on pages 54, 117, 156)

Cook, R. L., Porter, T., and Carpenter, L. (1984). Distributed ray tracing. In *Computer Graphics: Proceedings of SIGGRAPH '84*, Vol. 18, No. 3, pages 137–145.
(cited on pages 43, 54)

Crocker, G. A. (1984). Invisibility coherence for faster scan-line hidden surface algorithms. In *Computer Graphics: Proceedings of SIGGRAPH '84*, Vol. 18, No. 3, pages 95–102.
(cited on page 167)

Crow, F. C. (1977). The aliasing problem in computer-generated shaded images. *Communications of the ACM*, Vol. 20, No. 11, pages 799–805.
(cited on page 48)

Crow, F. C. (1981). A comparison of antialiasing techniques. *IEEE Computer Graphics and Applications*, Vol. 1, No. 1, pages 40–48.
(cited on page 52)

Crow, F. C. (1984). Summed-area tables for texture mapping. In *Computer Graphics: Proceedings of SIGGRAPH '84*, Vol. 18, No. 3, pages 207–212.
(cited on page 116)

Datapath (1991a). *Merlin Development Toolkit*. Datapath Ltd, Alfreton Road, Derby, DE2 4AD, UK. Version 1.01.
(cited on page 32)

Datapath (1991b). *Merlin User Manual.* Datapath Ltd, Alfreton Road, Derby, DE2 4AD, UK. Version 1.01.
(cited on page 30)

Davson, H. (1980). *Physiology of the Eye.* Churchill Livingstone, Edinburgh, London, and New York, fourth edition.
(cited on pages 12, 15)

Deering, M. (1992). High resolution virtual reality. In *Computer Graphics: Proceedings of SIGGRAPH '92*, Vol. 26, No. 2, pages 195–202.
(cited on page 66)

Devarajan, R. and McAllister, D. F. (1991). Stereoscopic ray tracing of implicitly defined functions. In *Stereoscopic Displays and Applications II*, Proc. SPIE 1457, pages 37–48.
(cited on pages 47, 90)

Dippé, M. A. Z. and Wold, E. H. (1985). Antialiasing through stochastic sampling. In *Computer Graphics: Proceedings of SIGGRAPH '85*, Vol. 19, No. 3, pages 69–78.
(cited on pages 54, 117)

Eichenlaub, J. B. (1994). An autostereoscopic display with high brightness and power efficiency. In *Stereoscopic Displays and Virtual Reality Systems*, Proc. SPIE 2177, pages 4–15.
(cited on pages 10, 77)

Ezell, J. D. and Hodges, L. F. (1990). Some preliminary results on using spatial locality to speed up ray tracing of stereoscopic images. In *Stereoscopic Displays and Applications*, Proc. SPIE 1256, pages 298–306.
(cited on pages 47, 48, 90, 124, 125, 126, 127, 129, 130)

Faris, S. M. (1994). Novel 3-D stereoscopic imaging technology. In *Stereoscopic Displays and Virtual Reality Systems*, Proc. SPIE 2177, pages 180–195.
(cited on page 6)

Fiume, E., Fournier, A., and Rudolph, L. (1983). A parallel scan conversion algorithm with anti-aliasing for a general-purpose ultracomputer. In *Computer Graphics: Proceedings of SIGGRAPH '83*, Vol. 17, No. 3, pages 141–150.
(cited on page 52)

Foley, J. D., van Dam, A., Feiner, S. K., and Hughes, J. F. (1990). *Computer Graphics: Principles and Practice.* The Systems Programming Series. Addison-Wesley Publishing Company, second edition.
(cited on pages 38, 41, 49, 59, 90, 94, 97, 106, 114)

Fuchs, H., Goldfeather, J., Hultquist, J. P., Spach, S., Austin, J. D., Brooks, Jr., F. P., Eyles, J. G., and Poulton, J. (1985). Fast spheres, shadows, textures, transparencies and image enhancements in Pixel-planes. In *Computer Graphics: Proceedings of SIGGRAPH '85*, Vol. 19, No. 3, pages 111–120.
(cited on pages 44, 55, 118)

Fuchs, H., Kedem, Z. M., and Naylor, B. F. (1980). On visible surface generation by a priori tree structures. In *Computer Graphics: Proceedings of SIGGRAPH '80*, Vol. 14, No. 3, pages 124–133.
(cited on pages 42, 46)

Glassner, A. S. (1984). Space subdivision for fast ray tracing. *IEEE Computer Graphics and Applications*, Vol. 4, No. 10, pages 15–22.
(cited on page 43)

Glassner, A. S. (1988). Spacetime ray tracing for animation. *IEEE Computer Graphics and Applications*, Vol. 8, No. 2, pages 60–70.
(cited on page 46)

Goodrich, M. T. (1992). A polygonal approach to hidden-line and hidden-surface elimination. *CVGIP: Graphical Models and Image Processing*, Vol. 54, No. 1, pages 1–12.
(cited on page 41)

Goral, C. M., Torrance, K. E., Greenburg, D. P., and Battaile, B. (1984). Modeling the interaction of light between diffuse surfaces. In *Computer Graphics: Proceedings of SIGGRAPH '84*, Vol. 18, No. 3, pages 213–222.
(cited on pages 93, 96)

Gordon, D. and Chen, S. (1991). Front-to-back display of BSP trees. *IEEE Computer Graphics and Applications*, Vol. 11, No. 5, pages 79–85.
(cited on page 162)

Gouraud, H. (1971). Continuous shading of curved surfaces. *IEEE Transactions on Computers*, Vol. C-20, No. 6, pages 623–629.
(cited on pages 93, 113)

Greene, N. and Kass, M. (1994). Error-bounded antialiased rendering of complex environments. In *Computer Graphics: Proceedings of SIGGRAPH '94*, Annual Conference Series, pages 59–66.
(cited on page 45)

Greene, N., Kass, M., and Miller, G. (1993). Hierarchical Z-buffer visibility. In *Computer Graphics: Proceedings of SIGGRAPH '93*, Annual Conference Series, pages 231–238.
(cited on pages 45, 162)

Grimes, J., Kohn, L., and Bharadhwaj, R. (1989). The Intel i860 64-bit processor: A general-purpose CPU with 3D graphics capabilities. *IEEE Computer Graphics and Applications*, Vol. 9, No. 4, pages 85–94.
(cited on page 30)

Guo, B., Yamamoto, T., and Aoki, Y. (1988). A method of fast creating hologram source pictures from single view image structure. *Transactions of the IEICE*, Vol. E71, No. 5, pages 530–538.
(cited on pages 47, 90, 124, 125, 126, 130, 167)

Haeberli, P. and Akeley, K. (1990). The accumulation buffer: Hardware support for high-quality rendering. In *Computer Graphics: Proceedings of SIGGRAPH '90*, Vol. 24, No. 4, pages 309–318.
(cited on pages 45, 55, 117, 118, 119, 156)

Halle, M. W. (1994). Holographic stereograms as discrete imaging systems. In *Practical Holography VIII*, Proc. SPIE 2176, pages 73–84.
(cited on pages 56, 57, 78, 167)

Harrison, L. and McAllister, D. F. (1993). Implementation issues in interactive stereo systems. In *Stereo Computer Graphics and Other True 3D Technologies*, McAllister, D. F., editor, chapter 7, pages 119–151. Princeton University Press, Princeton, New Jersey.
(cited on pages 47, 66, 126, 129, 130)

Heckbert, P. S. (1986). Survey of texture mapping. *IEEE Computer Graphics and Applications*, Vol. 6, No. 11, pages 56–67.
(cited on page 116)

Heckbert, P. S. and Moreton, H. P. (1991). Interpolation for polygon texture mapping and shading. In *State of the Art in Computer Graphics: Visualization and Modeling*, Rogers, D. F. and Earnshaw, R. A., editors, pages 101–111. Springer-Verlag, New York.
(cited on page 116)

Hodges, L. F. (1991). Basic principles of stereographic software development. In *Stereoscopic Displays and Applications II*, Proc. SPIE 1457, pages 9–17.
(cited on pages 64, 66, 69, 72, 73, 76)

Hodges, L. F. (1992). Tutorial: Time-multiplexed stereoscopic computer graphics. *IEEE Computer Graphics and Applications*, Vol. 12, No. 2, pages 20–30.
(cited on pages 46, 64, 89)

Hodges, L. F. and McAllister, D. F. (1993). Computing stereoscopic views. In *Stereo Computer Graphics and Other True 3D Technologies*, McAllister, D. F., editor, chapter 5, pages 71–89. Princeton University Press, Princeton, New Jersey.
(cited on pages 67, 69)

Hubschman, H. and Zucker, S. W. (1981). Frame-to-frame coherence and the hidden surface computation: Constraints for a convex world. In *Computer Graphics: Proceedings of SIGGRAPH '81*, Vol. 15, No. 3, pages 45–54.
(cited on page 46)

Isono, H., Yasuda, M., Takemori, D., Kanayama, H., Yamada, C., and Chiba, K. (1992). 50-inch autostereoscopic full-color 3-D TV display system. In *Stereoscopic Displays and Applications III*, Proc. SPIE 1669, pages 176–185.
(cited on page 77)

Kajiya, J. T. (1986). The rendering equation. In *Computer Graphics: Proceedings of SIGGRAPH '86*, Vol. 20, No. 4, pages 143–150.
(cited on pages 54, 93, 111, 117)

Kay, D. S. and Greenberg, D. (1979). Transparency for computer synthesized images. In *Computer Graphics: Proceedings of SIGGRAPH '79*, Vol. 13, No. 2, pages 158–164.
(cited on page 93)

Kay, T. L. and Kajiya, J. T. (1986). Ray tracing complex scenes. In *Computer Graphics: Proceedings of SIGGRAPH '86*, Vol. 20, No. 4, pages 269–278.
(cited on page 43)

Kernighan, B. W. and Ritchie, D. M. (1988). *The C Programming Language*. Prentice Hall software series. Prentice Hall, Englewood Cliffs, New Jersey, second edition.
(cited on pages 34, 138)

Kirk, D. and Arvo, J. (1991). Unbiased sampling techniques for image synthesis. In *Computer Graphics: Proceedings of SIGGRAPH '91*, Vol. 25, No. 4, pages 153–156.
(cited on page 54)

Lane, B. (1982). Stereoscopic displays. In *Processing and Display of Three-Dimensional Data*, Proc. SPIE 367, pages 20–32.
(cited on page 4)

Lang, S. R., Travis, A. R. L., Castle, O. M., and Moore, J. R. (1992). A 2nd generation autostereoscopic 3-D display. In *Proceedings of the Seventh Eurographics Workshop on Graphics Hardware*, Eurographics Technical Report Series, pages 53–63. Workshop held 5–6 September 1992, King's College, Cambridge, England.
(cited on pages 9, 77)

Lee, M. E., Redner, R. A., and Uselton, S. P. (1985). Statistically optimized sampling for distributed ray tracing. In *Computer Graphics: Proceedings of SIGGRAPH '85*, Vol. 19, No. 3, pages 61–67.
(cited on pages 54, 117)

Lipscomb, J. S. (1989). Experience with stereoscopic display devices and output algorithms. In *Three-Dimensional Visualization and Display Technologies*, Proc. SPIE 1083, pages 28–34.
(cited on pages 6, 66, 69)

Lipton, L. (1982). *Foundations of the Stereoscopic Cinema*. Van Nostrand Reinhold, New York.
(cited on page 114)

Lipton, L. (1987). Factors affecting "ghosting" in time-multiplexed plano-stereoscopic CRT display systems. In *True 3D Imaging Techniques and Display Technologies*, Proc. SPIE 761, pages 75–78.
(cited on page 16)

Lipton, L. (1993). Composition for electrostereoscopic displays. In *Stereo Computer Graphics and Other True 3D Technologies*, McAllister, D. F., editor, chapter 2, pages 11–25. Princeton University Press, Princeton, New Jersey.
(cited on page 77)

Mammen, A. (1989). Transparency and antialiasing algorithms implemented with the virtual pixel maps technique. *IEEE Computer Graphics and Applications*, Vol. 9, No. 4, pages 43–55.
(cited on pages 44, 55, 117, 118)

Margulis, N. (1990). *i860 Microprocessor Architecture*. Osborne McGraw-Hill, 2600 Tenth Street, Berkeley, California 94710, U.S.A.
(cited on page 30)

McAllister, D. F. (1992a). 3-D displays. *BYTE*, Vol. 17, No. 5, pages 183–188.
(cited on page 2)

McAllister, D. F. (1992b). On minimizing absolute parallax in a stereo image. In *Stereoscopic Displays and Applications III*, Proc. SPIE 1669, pages 20–30.
(cited on page 77)

McAllister, D. F., editor (1993). *Stereo Computer Graphics and Other True 3D Technologies*. Princeton University Press, Princeton, New Jersey.
(cited on pages 1, 4)

Milgram, P. and Krüger, M. (1992). Adaptation effects in stereo due to on-line changes in camera configuration. In *Stereoscopic Displays and Applications III*, Proc. SPIE 1669, pages 122–134.
(cited on page 77)

Mitchell, D. P. (1987). Generating antialiased images at low sampling densities. In *Computer Graphics: Proceedings of SIGGRAPH '87*, Vol. 21, No. 4, pages 65–72.
(cited on pages 54, 117)

Mitchell, D. P. (1991). Spectrally optimal sampling for distribution ray tracing. In *Computer Graphics: Proceedings of SIGGRAPH '91*, Vol. 25, No. 4, pages 157–164.
(cited on pages 54, 117, 118)

Naimark, M. (1991). Elements of realspace imaging: A proposed taxonomy. In *Stereoscopic Displays and Applications II*, Proc. SPIE 1457, pages 169–179.
(cited on page 124)

Newell, M. E., Newell, R. G., and Sancha, T. L. (1972). A solution to the hidden surface problem. In *Proceedings of the ACM National Conference*, pages 443–450.
(cited on page 42)

Nishita, T. and Nakamae, E. (1985). Continuous tone representation of three-dimensional objects taking account of shadows and interreflection. In *Computer Graphics: Proceedings of SIGGRAPH '85*, Vol. 19, No. 3, pages 23–30.
(cited on page 93)

Painter, J. and Sloan, K. (1989). Antialiased ray tracing by adaptive progressive refinement. In *Computer Graphics: Proceedings of SIGGRAPH '89*, Vol. 23, No. 3, pages 281–288.
(cited on pages 55, 117)

Papathomas, T. V., Schiavone, J. A., and Julesz, B. (1987). Stereo animation for very large data bases: Case study— meteorology. *IEEE Computer Graphics and Applications*, Vol. 7, No. 9, pages 18–27.
(cited on pages 47, 90)

Phong, B. T. (1975). Illumination for computer generated pictures. *Communications of the ACM*, Vol. 18, No. 6, pages 311–317.
(cited on pages 93, 95, 114)

Portland Group (1991). *PGTools Documentation*. The Portland Group, 9150 SW Pioneer Court, Suite H, Wilsonville, Oregon 97070. Release 2.0.
(cited on page 34)

Potmesil, M. and Chakravarty, I. (1981). A lens and aperture camera model for synthetic image generation. In *Computer Graphics: Proceedings of SIGGRAPH '81*, Vol. 15, No. 3, pages 297–305.
(cited on page 45)

Ratliff, F. (1965). *Mach Bands: Quantitative studies on neural networks in the retina*. Holden-Day, Inc., San Francisco, London, Amsterdam.
(cited on page 113)

Reinhart, W. F. (1992). Gray-scale requirements for anti-aliasing of stereoscopic graphic imagery. In *Stereoscopic Displays and Applications III*, Proc. SPIE 1669, pages 90–100.
(cited on page 55)

Robinett, W. and Rolland, J. P. (1991). A computational model for the stereoscopic optics of a head-mounted display. In *Stereoscopic Displays and Applications II*, Proc. SPIE 1457, pages 140–160.
(cited on page 66)

Rubin, S. M. and Whitted, T. (1980). A 3-dimensional representation for fast rendering of complex scenes. In *Computer Graphics: Proceedings of SIGGRAPH '80*, Vol. 14, No. 3, pages 110–116.
(cited on page 43)

Schilling, A. (1991). A new simple and efficient antialiasing with subpixel masks. In *Computer Graphics: Proceedings of SIGGRAPH '91*, Vol. 25, No. 4, pages 133–141.
(cited on page 52)

Schilling, A. and Straßer, W. (1993). EXACT: Algorithm and hardware architecture for an improved A-buffer. In *Computer Graphics: Proceedings of SIGGRAPH '93*, Annual Conference Series, pages 85–91.
(cited on page 52)

Sexton, I. (1989). Parallax barrier 3DTV. In *Three-Dimensional Visualization and Display Technologies*, Proc. SPIE 1083, pages 84–94.
(cited on page 1)

Sharir, M. and Overmars, M. H. (1992). A simple output-sensitive algorithm for hidden surface removal. *ACM Transactions on Graphics*, Vol. 11, No. 1, pages 1–11.
(cited on page 41)

Starks, M. (1991). Stereoscopic video and the quest for virtual reality: an annotated bibliography of selected topics. In *Stereoscopic Displays and Applications II*, Proc. SPIE 1457, pages 327–342.
(cited on page 4)

Starks, M. (1992). Stereoscopic video and the quest for virtual reality: an annotated bibliography of selected topics — part II. In *Stereoscopic Displays and Applications III*, Proc. SPIE 1669, pages 216–227.
(cited on page 4)

Sutherland, I. E., Sproull, R. F., and Schumacker, R. A. (1974). A characterization of ten hidden-surface algorithms. *Computing Surveys*, Vol. 6, No. 1, pages 1–55.
(cited on pages 40, 41, 42, 46, 111, 161)

Tessman, T. (1990). Perspectives on stereo. In *Stereoscopic Displays and Applications*, Proc. SPIE 1256, pages 22–27.
(cited on pages 64, 67, 69, 76)

Texas Instruments (1988). *TMS34010 User's Guide*. Texas Instruments Incorporated, Post Office Box 1443, Houston, Texas.
(cited on page 30)

Touris, T. C. (1994). System performance requirements for a head tracking autostereoscopic display. In *Stereoscopic Displays and Virtual Reality Systems*, Proc. SPIE 2177, pages 16–25.
(cited on page 10)

Travis, A. R. L. (1990). Autostereoscopic 3-D display. *Applied Optics*, Vol. 29, No. 29, pages 4341–4343.
(cited on pages 9, 10)

Travis, A. R. L. and Lang, S. R. (1991). The design and evaluation of a CRT-based autostereoscopic 3-D display. In *Proceeding of the Society for Information Display*, Vol. 32, No. 4, pages 279–283.
(cited on pages 9, 17)

Ward, G. J. (1994). The RADIANCE lighting simulation and rendering system. In *Computer Graphics: Proceedings of SIGGRAPH '94*, Annual Conference Series, pages 459–472.
(cited on pages 48, 90, 127, 129, 130)

Whitted, T. (1980). An improved illumination model for shaded display. *Communications of the ACM*, Vol. 23, No. 6, pages 343–349.
(cited on pages 42, 43, 93)

Williams, L. (1978). Casting curved shadows on curved surfaces. In *Computer Graphics: Proceedings of SIGGRAPH '78*, Vol. 12, No. 3, pages 270–274.
(cited on page 45)

Williams, L. (1983). Pyramidal parametrics. In *Computer Graphics: Proceedings of SIGGRAPH '83*, Vol. 17, No. 3, pages 1–11.
(cited on pages 116, 156)

Williams, S. P. and Parrish, R. V. (1990). New computational control techniques and increased understanding for stereo 3-D displays. In *Stereoscopic Displays and Applications*, Proc. SPIE 1256, pages 73–82.
(cited on pages 64, 69, 72, 73, 77)

Wiseman, N. E. (1990). Graphics II. Lecture course notes used at the University of Cambridge, Computer Laboratory, New Museums Site, Pembroke Street, Cambridge, UK.
(cited on page 64)

Yeh, Y.-Y. (1993). Visual and perceptual issues in stereoscopic colour displays. In *Stereo Computer Graphics and Other True 3D Technologies*, McAllister, D. F., editor, chapter 4, pages 50–70. Princeton University Press, Princeton, New Jersey.
(cited on pages 74, 77)