HYBRID IMAGE/MODEL BASED GAZE-CONTINGENT RENDERING

A Dissertation
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy
Computer Science

---

by
Hunter A. Murphy
December 2007

Accepted by:
Dr. Andrew T. Duchowski, Committee Chair
Dr. Robert M. Geist III
Dr. Timothy A. Davis
Dr. Richard A. Tyrrell

Abstract

A non-isotropic hybrid image/model based gaze-contingent rendering technique utilizing ray casting on a GPU is discussed. Empirical evidence derived from human subject experiments indicates an inverse relationship between a peripherally degraded scene's high-resolution inset size and mean search time, a trend consistent with existing image-based and model-based techniques. In addition, the data suggest that maintaining a target's silhouette edges decreases search times when compared to targets with degraded edges. Benefits of the hybrid technique include simplicity of design and parallelizability, both conducive to GPU implementation.

Acknowledgments

Special thanks belong to Dr. Duchowski for both his enthusiasm and patience in guiding me through this process. Without the knowledge and patronage he provided I would not have been able to complete this project.

I would like to thank my advisory committee for their constructive criticism, without which this dissertation would be diminished. Each member was selected specifically for the applicability of their research towards this project, and I have not been disappointed by my choices.

The Computer Science department in general deserves my thanks, as the strong foundation I received, in every topic, contributed to this project.

I would also like to thank my fellow grad students for showing me it can be done!

My friends, who often tease me about my extended educational convalescence, have my thanks for supporting me when the teasing was over. Whenever I needed an extra pair of eyes for testing, someone was available.

Additional thanks go to my family for their tireless support of my goals. My sister and brother-in-law have often served as guinea pigs in sometimes uncomfortable experiments. My parents have offered not only moral support, but have worked hard to make sure we always received the best education possible.

Finally, my wife Elizabeth Murphy deserves more credit than I can express for her role in my completion of this process. She has bolstered my spirits, helped direct my energy constructively, and kept me focused. Above all, she has been patient with me, sacrificing for the long term benefits of my education. We are a team.

Table of Contents

Table of Contents (Continued)

Page

Table of Contents (Continued)

Page

List of Figures

Chapter 1

Introduction

Real-time photorealistic rendering is an as yet unattained goal of the computer graphics community. Achieving photorealism has become possible through the use of advanced ray tracing techniques, but these methods are far from interactive. For researchers interested in immersive, interactive graphics (such as virtual reality), photorealism is sacrificed for the more important real-time component. Thus, for the most part, rasterization of polygonal meshes remains the only viable option for interactive graphics.

Although photorealism has yet to be attained in a *virtual environment* (VE), the desire for photo-realistic rendering remains strong. Perhaps the most obvious approach is to use geometric models sampled at a very high frequency. This results in non-interactive rendering, due to the fact that current rendering hardware has difficulty with the potentially millions of polygons in the resulting VE. Researchers have investigated various rendering approaches targeting features of the *human visual system* (HVS) in an effort to decrease the number of polygons required in any given frame.

It is known that visual acuity decreases as a function of the distance between target and viewpoint. This fact has been exploited through the use of isotropic degradation in the form of image-based and model-based mipmapping. However, visual acuity also decreases as a function of angular separation between the target and the fovea, or *point of regard* (POR). The resulting non-isotropic degradation techniques, generally referred to as *gaze-contingent rendering* (GCR), are necessarily more involved than their isotropic relatives, but can offer an additional opportunity to reduce peripheral complexity, and thus polygon counts.

Although model-based gaze-contingent rendering has proven feasible, reducing

3D geometry in an HVS acceptable manner is challenging. Image-based gaze-contingent rendering is much less complicated, but lacks the ability to reduce polygon counts for the model being rendered. The goals of this project are to develop a hybrid image/model based gaze-contingent rendering technique, thereby combining the best practices of both classes of rendering, and to empirically demonstrate the viability of the technique.

Chapter 2 introduces historical information pertinent to the general area of perceptually based graphics. Chapter 3 gives particular attention to gaze-contingent rendering, including an overview of the two main branches of inquiry (screen- and model- based rendering). Chapter 4 discusses the new ray-casting based hybrid approach, and describes the two human visual system factors which will be exploited in this experiment. The CPU and GPU implementation specifics are provided in Chapters 5 and 6 respectively. Chapter 7 presents the experimental methodology, and is followed by the results and conclusions (Chapter 8).

Chapter 2

Background

Many techniques exist to mislead the human visual system into believing that it has more data than is actually present. Texture mapping and bump mapping are common techniques used to increase apparent geometry through surface coloration and reflectivity redirection. Extreme cases, such as impostors, bypass the rendering of geometry entirely and consist of simple 2D images.

This paper focuses on a technique called gaze-contingent rendering. Obtaining a user's gaze direction at run-time allows much more advanced manipulation of the human visual system than simply rendering fake geometry; the models themselves can be altered to tightly adhere to human perception. Section 2.1 discusses some of the physical and perceptual characteristics of the human visual system, and Section 2.2 explains various eye tracking techniques. In addition, gaze-contingent rendering is introduced, prior to a more thorough treatment in the following chapter.

2.1 Physiology and Perception

Visual perception is based on the sampling rate that the human eye applies to its surroundings. Additional processing occurs as data travels to the brain, but no new real information is generated. Thus the physical configuration of the eye is vital to understanding and manipulating perception.

Of direct importance to the sampling rate of the human visual system is the distribution of rods and cones across the retina. Figure 2.1 shows the the abundance of rod and cone cells as a function of visual angle [Osterberg 1935]. It is clear from the graph that cone cells fall off sharply within 5 degrees of the fovea. The decrease in cone cells is accompanied by an increase in rod cells; a comparison of key features of both cell types is shown in Table 2.1.

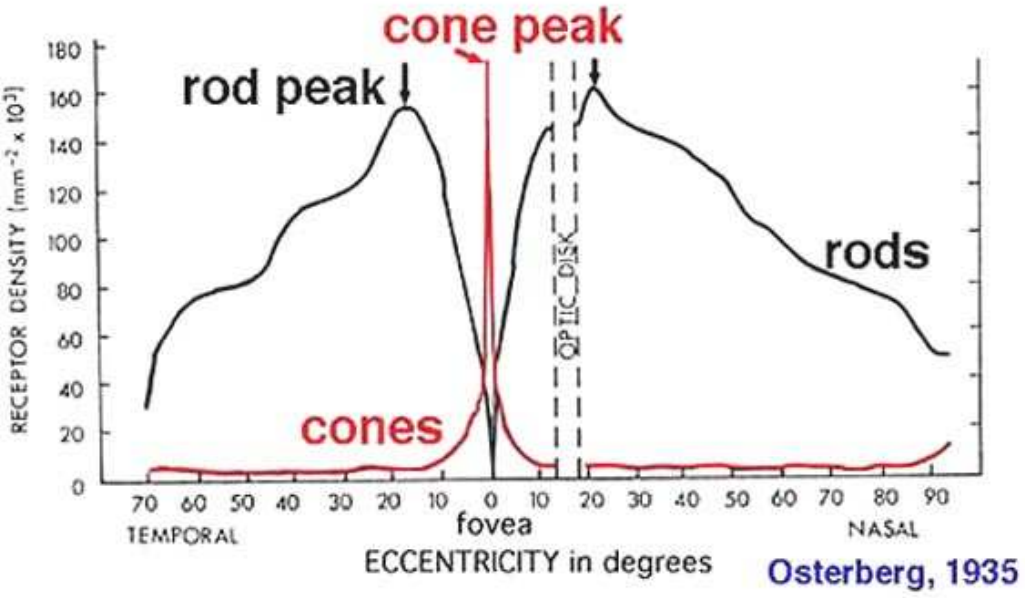

Figure 2.1: Distribution of retinal cells

The characterization of rods and cones directly reflects on the overall response of the human visual system to stimulus. Rods have slow response times and high sensitivity, indicating that rapid movements will be more visible in the periphery than near the fovea. They possess low acuity, resulting in an inability to resolve high spatial frequency detail. Their reduced ability to react to different wavelengths, coupled with the lack of directional sensitivity to incoming light, makes rods particularly sensitive to luminance. Cones, on the other hand, offer high acuity, rapid response times, and greater sensitivity to different wavelengths, making them ideal for absorbing high spatial frequency information.

Rods	Cones
Slow response: long integration time	Fast response: short integration time
Not directionally selective	Directionally selective
High sensitivity	Lower absolute sensitivity
Low acuity	High acuity
Achromatic: one type of pigment	Chromatic: three types of pigment

Table 2.1: Summary of Retinal Cell Characteristics

2.1.1 HVS Targets

The human visual system is highly complex, involving both the morphology of the eye and the physiology of the brain and nervous system. Given the physiological aspects discussed above, it is clear than many opportunities exist for manipulation of the human visual system. Some areas that have been exploited include:

1. Chromaticity
2. Kinetic
3. Luminance

Chromaticity

As previously mentioned, cone cells are more expressive of different frequencies of light than rods. The distribution of rods and cones results in relatively poor ability to discern color in the periphery. Researchers have attempted to exploit this reduction in chromatic response to reduce the amount of information (i.e., bits per pixel) required to represent peripheral colors [Reddy 1997; Duchowski and Çöltekin 2007].

Kinetic

A feature of rod cells is their relatively slow "refresh rate." While this does contribute to sensitivity to peripheral motion, the reduced visual acuity means that a moving object cannot be resolved as clearly as a stationary object. This has led to the targeting of peripheral motion for reducing the resolution of object interaction.

Collisions are one of the major types of physical interaction present in a simulation. Often a limiting factor in render times, collision modeling can benefit from human perceptual limitations. O'Sullivan et al. [2002] explored gaze-contingent LOD collision handling, a mechanism in which collision detection, contact modeling, and collision response are calculated based on eccentricity. This approach necessitates

the construction of a hierarchy of bounding volume approximations of an object and a method of indexing that hierarchy. O'Sullivan mapped eccentricity into a scheduler priority scheme, which allowed foveated objects more time to traverse the hierarchy, hence increasing the accuracy of the predicted collision.

Luminance

Luminance is widely exploited in perceptual rendering due to its relation to mesh geometry and contrast. For example, this paper utilizes idealized contrast sensitivity as a basis for screen sampling. For regions with high luminance, such as specular highlights, detail in the affected area can be washed out. This condition of reduced perceptibility, in which any detail in the region is replaced with uniformly saturated pixels, represents a target for perceptual degradation [Williams et al. 2003].

2.1.2 HVS Edge Coherence

Maintaining an object's silhouette edges while exploiting other facets of the human visual system is vitally important. The reasons are twofold: an inaccurate silhouette can introduce perceptually disruptive artifacts into an image, and silhouette edges are utilized by the human visual system to aid object characterization.

Image artifacts caused by silhouette inaccuracies can be particularly conspicuous to an observer; not only is the behavior unexpected for a static object, but the rapid oscillation in pixel coverage may actually draw the user's gaze to the problem area, exacerbating an already undesirable trait referred to as "popping." For example, if an LOD model is rendered at two different levels and the screen coverage is not identical, the condition shown in Figure 2.2a may result. In more extreme cases, mesh topology can be compromised (see last panel of Figure 3.1). These effects can be classified as aliasing artifacts.

Object recognition and categorization is also affected by silhouette edges. In a series of psychophysical experiments, Hayward [1998] explored the importance

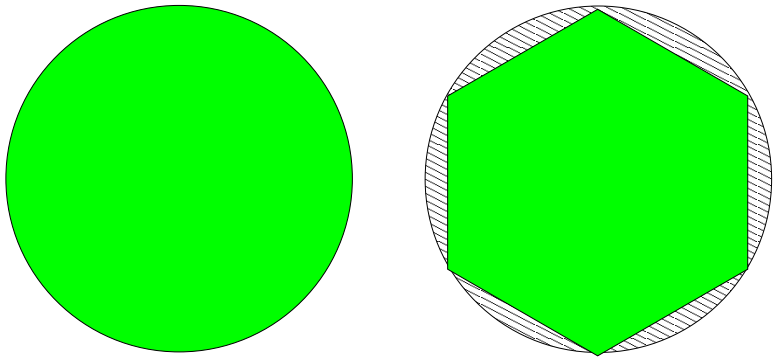

Figure 2.2: Popping can exist when moving between LOD levels with different screen coverage

of "outline shape", defined as the bounding contour of an object from a particular viewpoint. Human subjects were given two tasks: match an object to a shaded image or silhouette, and recognize a shaded image or silhouette. These tasks were repeated with the objects rotated between presentations to test viewpoint dependence. The results in each case indicated that changes to outline shape, or silhouette edge, predicted object recognition.

Hoffman and Singh [1997] assert that boundary strength is a strong indicator of salience. Their experiments focused on the *minima rule*, which indicates that human vision partitions objects at the negative minima of curvature on silhouettes. Human subjects were shown stimuli designed to test their interpretation of a simple pattern. Based on responses given to questions regarding the relative positions of a point in a scene to a simple pattern, Hoffman and Singh concluded that subjects routinely determined salience based on boundary strength.

Using eye tracking and nuclear magnetic resonance imaging, Vogels and Biederman [2002] measured neuron response to silhouette edges. Human subjects were not used (undoubtedly due to the requirement of surgical implantation of a "head post"); macaque monkeys were used instead. Measured neural activity indicated that the neurons imaged responded both to shaded and silhouetted-only objects, suggesting that the silhouette edge is sufficient for object selection for most neurons.

2.1.3 Perception Measurement Modality

Use of eye tracking equipment is at times impractical. The cost of acquiring a research grade eye tracker can easily extend to the tens of thousands of dollars. The effect being measured or exploited may not require decidegree accuracy. The technique being tested may not require real eye information at all. Any of these cases may employ alternative forms of evaluating the subject's gaze location.

A technique used by Ohshima et al. [1996] and Parkhurst et al. [2000] involved assuming that the subject's eye position always mapped to the center of the screen. Although clearly not realistic in the general case, given a situation in which the salient features can be forced to the center of the screen, the requirement for an eye tracker can be waived.

For situations in which more than one salient feature is present, an attentional model may be deployed. Attentional models (e.g., see O'Sullivan et al. [2003], Marmitt and Duchowski [2002], Itti [2004], Geisler and Perry [1998]) attempt to determine which regions a subject will foveate, and potentially the order of foveation. While reasonably accurate when integrated over a set of scanpaths, timing information such as fixation duration is not captured. However, attentional models can create saliency maps which can be used to guide vision.

Perceptual guidance attempts to force the subject to attend to particular regions at specific times. A common technique in marketing is to blur "unimportant" regions of an advertisement. The result of the masking is to emphasize low frequency information, driving attention to the "important" high frequency information. Another method of exploiting human vision is through motion. The sensitivity of the human visual system to peripheral motion can be used to guide attention between salient features simply by applying a short duration oscillation to the next salient feature to be viewed [Bailey et al. 2007].

2.2 Eye Tracking

The field of eye tracking can be separated into the hierarchy shown in Figure 2.3. Diagnostic use of eye tracking is frequently encountered in marketing research, where a customer's eye movements over time, i.e. scanpath, are recorded. Data compiled while viewing an advertisement can reveal vital information regarding colors, fonts, and placement of information. Diagnostic eye tracking is completely passive and typically involves post hoc analysis. Interactive eye tracking, however, attempts to use real-time data regarding the subject's gaze location to perform a task.

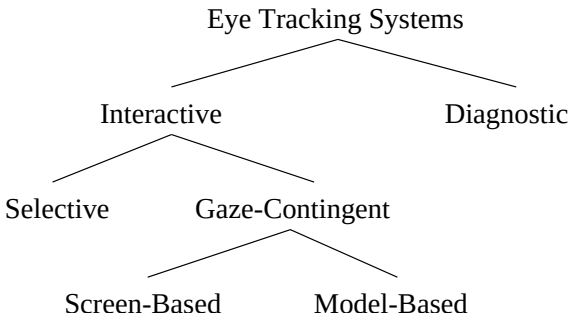

Figure 2.3: Hierarchy of eye tracking applications

Interactive eye tracking can be partitioned into two areas of research: selective and gaze-contingent. Selective eye tracking seeks to use gaze information as input similar to that provided by a mouse or joystick. By simply looking at a location on the screen, a user can move, activate, or otherwise manipulate data. Early work, as exemplified by Jacob, involved direct replacement of a mouse with an eye tracker. As research began to uncover some of the difficulties of using the eye for motor control tasks (see Jacob [1995] for a discussion of the "Midas Touch" problem), blended approaches were investigated in which the full burden of selection was not placed on the eye alone [Tanriverdi and Jacob 2000].

Gaze-contingent eye tracking uses real time gaze information to guide rendering in a perceptually appropriate manner. This typically involves reducing the amount of data rendered peripherally with respect to the point of regard. Image-based

gaze-contingent rendering attempts to achieve peripheral degradation through manipulation of the framebuffer contents immediately prior to display. Filtering is applied in an attempt to homogenize pixels, which can allow increased compression rates for display or transmission of the image. Model-based rendering attempts to remove geometry prior to rendering, which can decrease render times dramatically.

2.2.1 Foveation vs. Attention

It is important to note that eye tracking measures the point of regard, *not* necessarily the region attended to. While abrupt onset, for example rapid movement in the visual periphery, has long been known to induce coordinated eye/attention movement, Hunt and Kingstone [2003] investigated the relationship under volitional circumstances. Their research showed that deliberate eye movement can be decoupled from visual attention. This is important to gaze-contingent rendering because peripheral degradation can introduce artifacts. While tracking eye position can compensate by rendering with higher fidelity if the user's gaze is drawn to the artifact, visual attention can not be captured, resulting in a situation in which a deliberate user can discern degradation that is below the threshold of detection (with respect to eye position).

2.2.2 Eye Tracking Hardware

Two main categories of eye measurement technique exist: contact and non-contact. The contact methods are predominantly used to study the physiology of eye movements; specifically, to explore the neuromuscular substrate guiding visual attention. The non-contact methods are typically used in an attempt to discern visual attention through eye position, with goals external to the subject (*where* is the subject looking, rather than *how* is the subject looking).

The contact methods can be very accurate; however, as the name implies, they can be very invasive, sometimes involving physical contact with the eyeball itself.

For example, use of a scleral coil requires that the subject wear a contact lens with an embedded wire (see Figure 2.4a). By placing the subject's head in two orthogonal oscillating magnetic fields, induction in the wire can indicate coil position. It should be noted that this technique presents risks of corneal abrasion and disease transmission.

Electrooculography is a less invasive contact method which measures changes in voltage between the cornea and retina [Joyce et al. 2002]. Electrodes placed on the skin around the eye (Figure 2.4b) detect the rotation of the retino-corneal dipole axis, which can be interpreted as gaze direction. Although not as accurate as the scleral coil technique, electrooculography can obtain meaningful measurements over a much wider visual angle ($\pm 70^{\circ}$).

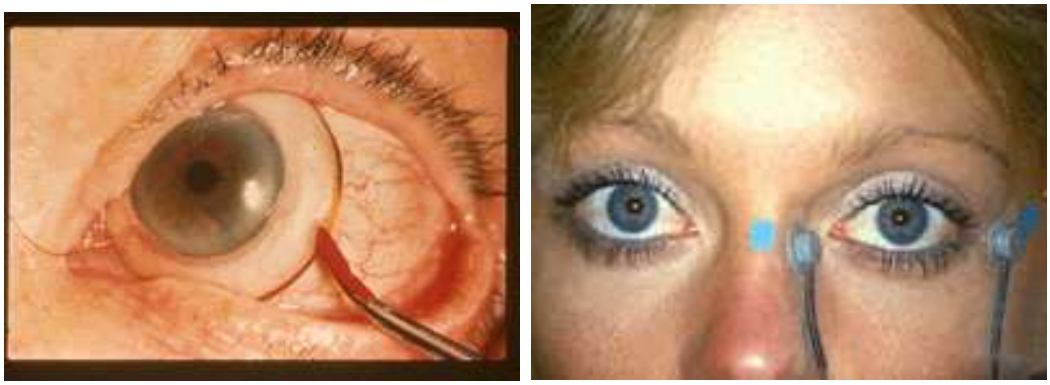

Figure 2.4: a. Scleral coil (left); b. Electrooculography (right)

Non-contact techniques can offer an unobtrusive alternative to the aforementioned eye tracking approaches; aside from projecting light outside the human visible spectrum, the techniques are entirely passive. One example is Dual-Purkinje image tracking, which measures the light reflected from the outside of the cornea and the inside of the lens. Infrared light is used to generate the reflections, called the first and fourth Purkinje images respectively (see Figure 2.5), and the difference between the two is used to calculate the subject's gaze. Obtaining a good fourth Purkinje image requires strict lighting control, as the reflection tends to be weak.

Video-based eye tracking, which utilizes pupil and corneal reflections, is another non-contact technique. By comparing the relative positions of the glint (first Purkinje image) and the retinal reflection, gaze direction can be determined. Both spatial and temporal resolution are good, but the technique is restricted to approximately 30° by the limited spherical region of the cornea.

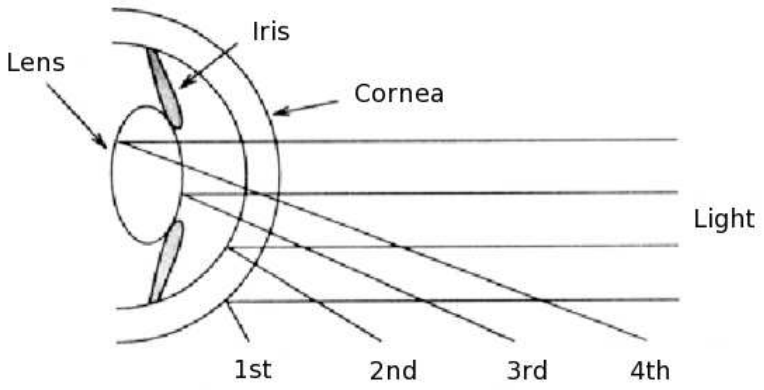

Figure 2.5: Diagram of Purkinje images

2.2.3 Gaze-Contingent Rendering

Gaze-contingent rendering exploits the human visual system's decreased peripheral sensitivity to several visual attributes. The goal in performing this analysis is to optimize visual fidelity and rendering performance. The measurement of performance can be ambiguous, but usually falls into two categories: functionally imperceptible degradation and visually imperceptible degradation.

Visually imperceptible degradation attempts to reduce peripheral information in a manner which cannot be distinguished from a non-degraded version. This stringent approach requires strict adherence to all aspects of the human visual system simultaneously. Use of visual imperceptibility as a performance metric is typically restricted to evaluation of the human visual system itself; e.g. experimental determination of a function describing the permissible extent of chromatic degradation in the periphery.

Functionally imperceptible degradation takes a more practical approach by interpreting imperceptible degradation to mean that the reduction in peripheral information does not result in reduced task performance. This less rigid stance allows much greater flexibility in manipulating the human visual system as it is focused on exploiting, rather than deriving, fundamental human visual system knowledge.

2.3 Image-Based GCR

Image-based gaze-contingent rendering is particularly well suited to exploring the perceptual limits of the human visual system. Implementation typically involves application of a convolution filter to a pre-rendered full resolution image. This simplifies not only the programming aspect, as 2D filtering is relatively simple to perform, but also the experimental aspect, since starting with an ideal image allows the experimenter to focus on manipulating a single perceptual variable of the human visual system.

2.4 Model-Based GCR

Model-based gaze-contingent rendering can significantly improve render time by removing mesh geometry which is not perceived by the user. The reduced complexity of geometry to be rendered is either generated from the original mesh through a series of edge collapses or vertex decimations, or is built up from a base mesh until it satisfies the requirements of the human visual system. Details regarding both image-based and model-based gaze-contingent rendering are provided in the following chapter.

Chapter 3

Related Work

This section contains brief descriptions of research directly applicable to the hybrid technique. The work of these selected authors covers image-based gaze-contingent rendering, model-based gaze-contingent rendering, and alternative techniques.

3.1 Image-Based GCR

3.1.1 Perceptual Research

Image-based gaze-contingent rendering exploits many features conducive to researching the human visual system. For image-based research the work of Watson et al. [1997] is particularly relevant. The authors studied the effects of LOD-based peripheral degradation on visual search performance. Both spatial and chrominance detail degradation effects were evaluated in head mounted displays. To sustain acceptable frame rates, two polygons were texture mapped in real-time to generate a high resolution inset within a low resolution display field. The authors suggested that spatial and chrominance complexity can be reduced by almost half without degrading performance.

Watson et al. [2004] performed an investigation into the validity of threshold-based LOD management in controlling supra-threshold information. They conjectured that LOD management which removed high frequency detail as a function of eccentricity may be counterproductive under certain circumstances, in particular low-contrast high-eccentricity conditions. To test this theory, a head tracked HMD with a $30^\circ \times 30^\circ$ high resolution inset coinciding with the center of the screen was used in place of an eye tracker (see Watson et al. [1997] and Watson et al. [1997]).

The periphery was degraded by rendering the entire scene into a small viewport, then mapping the viewport to the entire screen.

Filtering to construct the image-based LOD resulted in 5 discrete resolutions, varying from 0 to 50% of the frequency of the high resolution inset. No attempt was made to vary resolution dynamically with eccentricity; a preselected fixed resolution was selected from the aforementioned set. Blurring was performed at the interface between low and high resolution to soften the transition. Based on human subject experiments, Watson concluded that, beyond a threshold determined on a per-task basis, contrast is a better indicator of perceptibility than primitive size.

Geisler et al. [2006] conducted experiments to compare human visual search with that of a Bayesian ideal observer. The comparison to an ideal observer allowed the authors to determine the extent to which peripheral degradation increases visual search time. The search task consisted of localizing Gabor targets (sine wave gratings) against a background of noise which varied inversely with spatial frequency. The targets were constructed at four levels of spatial frequency, and the four backgrounds tested were created at varying root mean square values of noise contrast with the fixed background luminance. Peripheral degradation was controlled by scaling a falloff function; 6 discrete falloff values were tested.

The ideal observer implemented the following strategy: check multiple regions around the fixation point in parallel to determine the "best" match, if no matches exceed a specified criterion the fixation point shifts to the region that will provide the most information, at which point the process repeats. Human visual search compared very favorably with the ideal case. The authors suggest that human performance indicates that searches must be able to efficiently parallel process information, efficiently select optimal fixation locations, and inhibit fixations that would duplicate information. Additionally , Geisler et al. corroborated prior research indicating that, although reported as unnoticeable, peripheral degradation can affect search performance (presumably by reducing the ability to select fixations).

3.1.2 Bandwidth Utilization

By filtering out high frequency information in the periphery, image-based gaze-contingent rendering can offer a performance enhancement. The increase in image compressibility gained through peripheral degradation can be exploited to improve bandwidth utilization for image transmission [Bergström 2003].

Itti describes a technique for compressing video in response to human visual system parameters (see Itti [2004]). A dynamic saliency map is computed for the video clip as a whole using a combination of 12 features of the human visual system exhibited at the neuron level. The use of a whole-clip approach attempts to mimic vision without saccadic motions, which would negatively impact standard video compression algorithms (MPEG-1 and MPEG-4 were used to perform compression). Every pixel in each frame is then remapped to a restricted dynamic range through the use of a fourth order polynomial. The sequence of frames is then re-encoded.

To test the validity of various saliency map construction techniques, Itti compared the predicted regions of interest to real data from human subject experiments, in which the users viewed the unfoveated videos. Although the results were significantly similar in many cases, videos with motion or flicker cues generated the strongest agreement between the predictive model and the human subjects. The foveated videos achieved compressions ratios between 1.1 and 8.5 times greater than traditionally compressed videos.

While sophisticated approaches have been developed for *region of interest* (ROI) based image and video coding [Duchowski et al. 2004; Parkhurst and Niebur 2002; Reingold et al. 2002; Geisler et al. 2006], real-time computational benefits of image-based gaze-contingent rendering are lacking. Unless used for remote display, there can be no improvement in render time when the first step is to render the scene in full resolution, and the second is to apply a full screen convolution filter. Indeed, outside the uses in human visual system research and image transmission, image-based gaze-contingent rendering offers no practical benefit per se.

3.2 Model-Based GCR

3.2.1 Traditional LOD Rendering

Simplification of geometric objects to reflect perceptual limits is a widely used technique. One favored technique is to exploit the degradation of the human visual system as a function of distance from the viewer. First described by Clarke [1976], traditional *level of detail* (LOD) rendering utilizes the fact that the human visual system has a fixed sampling rate and is perspective based. As distance increases the visual angle subtended by the object decreases, resulting in a decreased sampling rate of that object. Since the human visual system is unable to directly perceive geometry sampled beyond that rate, that excess detail can be removed from the model [Reddy 1997]. Typically this geometric decimation occurs as a preprocessing step at predetermined distances (levels)(see Figure 3.1 adapted from Schaufler and Stürzlinger [1995]), though significant work in continuous LOD has been performed [Eck et al. 1995].

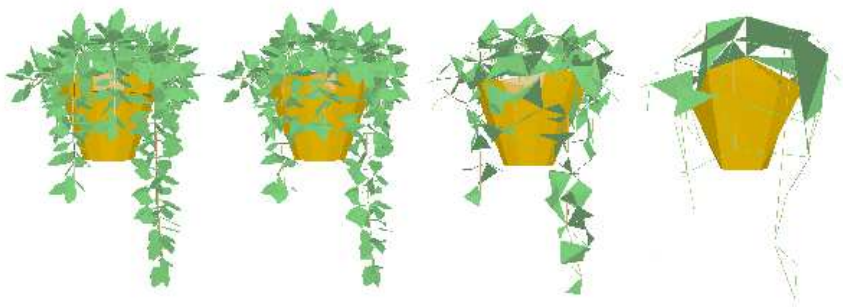

Figure 3.1: Traditional level of detail (from Schaufler and Stürzlinger [1995])

Unfortunately, the use of a hierarchy of models necessitates much larger storage space if it hopes to be of practical benefit. Additionally, isotropic degradation must not decimate geometry to a frequency lower than that required by the fovea, resulting in marginal benefits for high frequency meshes which subtend large visual angles.

LOD rendering has also been used in model-based gaze-contingent rendering.

Ohshima et al. [1996] proposed a scheme wherein three visual characteristics were considered: central/peripheral vision, kinetic vision, and fusional vision. The LOD algorithm generated isotropically degraded objects at different visual angles. Although the use of a binocular eye tracker was proposed, the system as discussed used only head tracking.

Reddy [1998] used a view-dependent LOD technique to evaluate both perceptual effects and system performance gains. LOD level selection was determined on a per-mesh basis by examining a table of precomputed spatial frequencies to estimate the instantaneous frequency of the mesh. The table was indexed by computing the mesh's spatial frequency from its screen coverage, angular velocity, and eccentricity.

Parkhurst and Niebur [2004] examined both velocity-based and gaze-contingent LOD in an attempt to gauge the applicability to consumer desktop computer systems. The discrete LOD levels were generated through an edge collapse mechanism, in which edges were iteratively collapsed based on an edge length error metric. Comparison between meshes with arbitrary geometric density, topology, and size, was permitted by normalizing the LOD levels against the original geometry. The error generated in the edge collapse process was accumulated and compared to the total edge lengths of the original mesh. The geometry was then rendered using a commercial game engine.

In their gaze-contingent LOD experiment, Parkhurst and Niebur varied the LOD level as a function of eccentricity. LOD degradation rates ran from 0.0 LOD units per degree to 0.2 LOD units per degree, and were applied linearly with increasing eccentricity. Degradation was clamped at 0.1 LOD units, presumably to reduce distracting peripheral artifacts induced by the edge collapse algorithm.

3.2.2 Multiresolution Geometric Modeling

Traditional LOD is clearly beneficial when rendering for a standard display paradigm; the use of isotropic object degradation for gaze-contingent rendering is suboptimal.

Uniform mesh degradation assumes uniform perceptual discrimination across the entire field of view. In this case, traditional LOD schemes will display a mesh at its full resolution even though the mesh may cover the entire field of view. Since acute resolvability of human vision is limited to the foveal 5°, object resolution need not be uniform within a gaze-contingent context. This is the central tenet of gaze-contingent systems. A system within which local mesh density at render time is contingent on gaze direction is called non-isotropic gaze-contingent rendering.

Traditional LOD systems also suffer from perceptual shortcomings, with respect to gaze-contingent rendering. Attempting to naively render the discrete levels of an LOD mesh may result in aliasing, or "popping." The visible artifacts of aliasing are caused by the often wide separation in screen coverage when rendering different LOD levels. The fundamental issue is that traditional LOD offers a very coarsely sampled hierarchy with no way to continuously represent intermediate resolutions. "Popping" in a gaze-contingent system is particularly noticeable, as eccentricity based LOD selections tend to change levels more rapidly than distance based selection.

Multiresolution geometric modeling attempts to address some of these issues. A paper by Hoppe [1997] successively applies wavelet transforms to a parametrically well-defined mesh and stores the coefficients. Compression of the data was achieved by culling coefficients that did not substantially affect the mesh geometry. Although the computational cost precluded direct use of their algorithm for non-isotropic gaze-contingent rendering, it did provide a starting point [Murphy and Duchowski 2001].

Multiresolution mesh modeling techniques suitable for gaze-contingent viewing have been developed [Zorin and Schröder 2000]. Techniques range from multiresolution representation of arbitrary meshes to the management of LOD through peripheral degradation within an HMD, where gaze position is assumed to coincide with head direction [Lindstrom et al. 1996; MacCracken and Joy 1996; Zorin et al. 1997; Schmalstieg and Schaufler 1997]. Multiresolution modeling allows the possibility of rendering meshes that are non-isotropically degraded by generating or selecting

localized geometry to display based on the angle subtended from the point of regard. Run-time construction of mesh geometry an be accomplished subtractively or additively.

3.2.3 Mesh Decimation

One technique for reducing mesh geometry is decimation. Starting with a full resolution mesh, primitives are removed when deemed to be unimportant to rendering. This technique is similar to back-face culling, in which non-visible geometry is removed, but applies to potentially visible primitives.

Luebke et al. [2000] and Luebke and Hallen [2001b] presented a technique based on vertex folding, which involved decimating a mesh by removal of perceptually redundant vertices and their connected triangles. The initial preprocessing required that the mesh be converted into a vertex tree, in which leaf nodes correspond to actual vertices, intermediate nodes correspond to clusters of vertices, and the root node represents the entire mesh. Each node contains a proxy vertex which acts as an average of all vertices represented in the node. Decimating a node in the tree results in all vertices supported by that node being replaced with the proxy vertex. It was to this existing library (VDSlib) that Luebke and Hallen added their perception criteria.

Perceptual simplification was achieved by developing an empirical model of a *contrast sensitivity function* (CSF). Subjects underwent a calibration phase to determine sensitivity to contrast gratings at varying spatial frequencies and eccentricities. The results were loaded into a look-up table which was indexed to determine if folding a particular node in the vertex tree would result in a perceptible change in contrast. The spatial frequency index was calculated by estimating the lowest spatial frequency effected by a vertex fold, which was determined by measuring the screen extent of a sphere bounding the vertex node. The contrast index was determined by comparing the total intensity of all vertices supported by the vertex node

with the vertices in the corresponding region of the original mesh. Special criteria were used to identify and preserve silhouette edges.

Danforth et al. [2000] developed a non-isotropic gaze-contingent multiresolution terrain navigation environment. A surface, represented as a quadrilateral mesh, was divided into fixed-size (number of vertices) sub-blocks, allowing rendering for variable LOD on a per-sub-block basis. Resolution level was chosen per sub-block, based on viewer distance. The resolution level was not discrete; it was interpolated between the pre-computed discrete levels to avoid "popping" effects. The approach used is reasonably effective; however, it is not clear whether the technique is applicable to arbitrary meshes.

3.2.4 Mesh Reconstitution

Rather than remove geometry from a full resolution representation, reconstitution attempts to create a new mesh from a severely decimated base mesh and information regarding the original mesh shape. This can be accomplished with as little as a single tetrahedron and a series of wavelet coefficients (obviously this depends on the topological constraints of the original mesh).

Murphy and Duchowski [2001] took this alternate approach by reconstituting the displayed mesh. This was accomplished by applying a hierarchy of subdivisions to a base mesh. The technique required significant preprocessing, but yielded an order of magnitude speedup in render time. The first step geodesically partitioned the original mesh geometry into Voronoi diagrams. Each Voronoi diagram was transformed into a 2D representation through the use of harmonic maps. Essentially, the corner vertices of the Voronoi diagram were anchored to a circle while each edge within the mesh was treated as a spring. After an energy minimization function was applied, the resulting 2D mesh was tessellated with triangles. Each triangle was recursively subdivided to an arbitrary depth in a parametrically uniform manner. The newly created 2D approximation was then "lofted" into 3D. First, the barycen-

tric coordinates of each vertex in the 2D approximation was calculated with respect to the harmonically mapped geometry. Those coordinates were then translated to the original 3D geometry using the same barycentric coordinates. The result of the remapping process was a 3D base mesh with a hierarchy of increasingly accurate refinements.

Reconstitution of the mesh was dictated by the spatial separation from the point of regard. Each triangle in the base mesh was tested against an acuity-based resolution degradation function. If any part of a triangle fell within the currently tested range of the degradation function, it was replaced with its next level children. The process was repeated for all levels of the degradation function.

3.3 Uncategorized Work

The aforementioned rendering techniques involve either image manipulation, or polygon rasterization, both standards in traditional rendering. Alternative techniques, however, have shown great promise in the realm of gaze-contingent rendering. Described below are splatting, which utilizes a point as a rendering primitive, and ray casting, which constructs an image based on intersections with scene geometry.

3.3.1 Splatting

Splatting is a rendering technique that departs greatly from traditional polygon rasterization. A point is used as the base geometric primitive, and is rendered as either a pixel or a 2D image centered on the point. An example is Rusinkiewicz and Levoy's [2000; 2001] QSplat infrastructure, originally used to display very large datasets (specifically noted is the output from 3D scanners, consisting of 100's of millions of points) from "The Digital Michelangelo Project."

QSplat first performs a preprocessing step in which the original geometry is converted into a sphere-based *bounding volume hierarchy* (BVH). Leaf spheres are

generated with a radius equal to the maximum length of any connected edge, a conservative metric used to guarantee complete mesh coverage. A recursive top-down approach is then taken to generate the hierarchy, with the dominant axis of the bounding box bisected and the leaves in each subdivided region driving the creation of the child bounding volumes.

Given the bounding volume hierarchy, culling is trivially implemented as an inside/outside test with the viewing frustum. Back-face culling is optionally implemented according to a test with a normal cone constructed during preprocessing. Rendering proceeds by rendering a splat when a leaf node is encountered, or when the screen area of the splat falls below the desired cutoff. Internal nodes exceeding the threshold are recursively traversed. It is important to note that frame rate was the metric used to determine the threshold, not visual fidelity.

Splats are rendered as 2D images with the same screen extent as the node being represented. Several approaches to drawing splats can be taken, but a "fuzzy spot" with radial alpha value decay was selected.

This technique is very similar to Luebke and Hallen's work with VDSlib; in fact, Luebke and Hallen adapted QSplat to include perceptual rendering [Luebke and Hallen 2001a]. The modification to QSplat was minimal, as the bounding spheres of the bounding volume hierarchy were directly used to determine contrast sensitivity, and the normal cones were directly used to determine silhouette edges (and given a value of maximum contrast). Rendering proceeded identically with QSplat's original model, except that the threshold used to decide whether a node's children should be traversed was guided by the computed contrast of rendering the node, rather than frame rate.

3.3.2 Ray Based Techniques

Levoy and Whitaker [1990] discuss a perceptually driven ray casting system for volumetric data. Traditional rendering of volume data involves the accumulation

of voxel information along a ray cast into the data set. Rays are typically cast uniformly, with uniform sampling along their length. Levoy and Whittaker instead sampled data sets non-uniformly, both in terms of visual angle and distance. The scene was sampled in three separate regions: a low resolution background, a high resolution inset, and a transitional region of variable resolution. The central region was ray cast at one ray per pixel, while the periphery was sampled at one ray per 16 pixels. The sampling rate of the intermediate region varied in an attempt to blur the interface between high and low sampling rates. Ray density in the intermediate region was determined by casting multiple rays through an area for which a representative color was needed. The exact number of rays required was calculated based on separation between the area being rendered and the point of regard, which was used as an index into a 2D mipmap of ray densities based on a simple Gaussian falloff. The desired number of rays were cast and their results integrated and filtered to produce a color value.

The rate of data sampling along the length of the ray was also based on visual perception. As distance between the eye point and the volumetric data increased, the sampling rate was reduced through the use of a precomputed 3D mipmap of the data set.

Dayal et al. [2005] introduced a concept called adaptive frameless rendering. This technique is perceptually based, with screen updates dependent on rate of change in luminance, but also takes into account a temporal component. This allows static regions to retain prior information while displaying the most recent samples for regions with high spatio-temporal rate of change. Although not mentioned, gaze-contingent rendering could both benefit and augment the system as described.

The algorithm involves maintaining a buffer of samples in both space and time, and filtering the samples (again, in both space and time) to provide a display image. Rays are cast into a scene, as with traditional ray tracing, but rather than cast rays in a rasterization like scheme, adaptive frameless rendering drives ray casting

through the use of a luminance and time gradient. This combined gradient drives ray densities to regions that are undergoing rapid changes. At any moment in time, a decoupled buffer can be filtered to provide a screen image.

This process may benefit from gaze-contingent feedback by offering additional attentional input to determine where rays are cast. Gaze-contingent rendering could certainly benefit from use of adaptive frameless rendering, as latency is often an issue with eye tracking input modalities.

Summary

Model-based rendering typically requires significant processing, either as a preprocessing step or at run-time [Luebke and Erikson 1997], but can provide a net speedup in rendering by removing geometry sampled at a rate greater than perceptually required. Non-isotropic rendering attempts to remove even more geometry. The original mesh is either decimated to a lower complexity through a series of edge collapses, or built up from a base mesh to a higher complexity representation.

Unfortunately, model-based gaze-contingent rendering lacks the simplicity of experimental and programmatic implementation inherent to image-based gaze-contingent rendering. Reconstruction of mesh geometry, either through decimation of the original mesh or reconstitution of a base mesh, is an issue of local mesh connectivity; however, maintaining the global perceptual requirements makes the utilization of model-based gaze-contingent rendering a non-trivial task.

Alternative methods, such as those based on ray casting or splatting, offer intriguing possibilities. Mesh connectivity is not a prohibiting factor in these techniques, nor is "correctness" of the resulting image, as areas of concern can be resampled at any rate desired by the user.

Chapter 4

Hybrid Technique

The goal of the hybrid technique is to non-destructively sample scene geometry in a manner consistent with the limits of the human visual system. This is achieved using ray casting, with ray distribution conforming to the angular frequency dictated by a *contrast sensitivity function* (CSF). The ray casting and contrast sensitivity function combination allows non-isotropic (within mesh) degradation without directly manipulating mesh geometry (edge collapse or vertex decimation). An intermediate mesh existing between the eye position and the scene geometry provides both a direction vector for ray casting and storage for the resulting ray/primitive intersection data. Further refinement of the intermediate mesh is performed to maintain silhouette edges. The intermediate mesh is ultimately rendered in place of the scene geometry.

4.1 Comparison with Image-Based GCR

The advantages of image-based gaze-contingent rendering stem from restricting all operations to two dimensions. For example, 2D differencing operations, such as those used to determine silhouette edges, are simple comparisons between adjacent pixels. The equivalent 3D operation involves a much more complex process [Sun et al. 2002]. Extending this example to gaze-contingent rendering, the analysis of a 3D scene to constrain human visual system parameters to an acceptable range requires significant ingenuity and computational effort. The extent of these difficulties often results in compromises to the visual fidelity of the resulting image.

The hybrid technique involves the construction of a “virtual retina”, an intermediate mesh restricted to two dimensions. Although the structure is a true mesh consisting of point data and connectivity information, it can also be treated as a

sparsely populated image. Analysis that would be difficult in 3D, such as a differencing operation, can proceed trivially; for example, the edge detection operation mentioned above is easily accomplished by taking the difference in values between adjacent data points. More precise edge detection can be achieved by localized high frequency re-sampling of the mesh.

4.2 Comparison with Model-Based GCR

Although more complex than image-based gaze-contingent rendering, the possibility of rendering dense meshes at interactive frame rates makes model-based gaze-contingent rendering an attractive goal. The decrease in render time ascribed to model-based gaze-contingent rendering is made possible by reducing the amount of geometry rendered. It is the selection of geometry to remove, based on the effect to the rendered image, that makes model-based gaze-contingent rendering non-trivial.

The hybrid technique offers the possibility of increased frame rate by partially decoupling the render time from the number of polygons in the mesh. Unlike model-based approaches, the geometry rendered by the hybrid technique is not decimated. Instead, the original geometry is sampled at a rate less than the density of the mesh primitives (dependent on eye/mesh separation). While it is true that external silhouette edges can increase render time by requiring locally higher sampling rates, the number of rays cast is dictated more by interaction between the screen and the human visual system than by polygon count. It is obvious that this relationship does not hold for low density scenes.

4.3 Parallelizable

Model-based rendering is an inherently serial operation. The nature of the geometric reduction, whether based on decimation or reconstitution, requires evaluation of each primitive's contribution to the final image. The applicability of parallelization

is thus very limited for model-based gaze-contingent rendering.

One feature in particular makes image-based gaze-contingent rendering a good candidate for parallel rendering: single pixel primitives which embody extremely localized coupling. Model-based gaze-contingent rendering typically operates on polygon primitives which can subtend large numbers of pixels. The decision to deselect a particular piece of geometry for rendering can therefore highly impact the resulting image. Removal of a single triangle can introduce changes which propagate through several levels of a multiresolution hierarchy. Image-based gaze-contingent rendering, however, operates on highly localized pixel groups. An image convolution kernel may only involve eight neighboring pixels in determining if a target pixel needs to be modified. This characteristic allows image-based gaze-contingent rendering to be performed simultaneously across the entire image.

The hybrid technique mimics the behavior of image-based gaze-contingent rendering by totally decoupling each ray from its neighbor during ray casting, allowing embarrassingly trivial parallelization. The only coupling of data results from refinement of the intermediate mesh, in this case edge detection, for which neighboring rays are queried in a manner analogous to the image convolution kernels used in image-based gaze-contingent rendering. Given the similarity to image-based gaze-contingent rendering, it is not surprising that the hybrid technique can take advantage of the massively parallel design of modern graphics processors.

4.4 Implementation

The following subsections discuss the implementation of the hybrid technique at a generic level and can be broken into two categories: ray casting and ray generation. The ray casting sections discuss some of the capabilities required for efficient operation. The ray generation sections focus on the perceptual requirements guiding the scene sampling rate and the methods used to translate those requirements into

rays. Specific details regarding algorithm selection and coding issues are postponed to the chapters on CPU and GPU implementation.

4.4.1 Ray Casting Acceleration Structures

In order to achieve efficient render times for any non-trivial scene it is necessary to reduce the number of ray/primitive intersection tests. The most common approach is to use a ray casting acceleration structure, which places boundaries on the search space to be traversed when determining if an intersection has occurred [Arvo and Kirk 1989]. Modern ray casting acceleration structures fall into two bodies of solutions: bounding volume hierarchies and spatial subdivision.

Bounding Volume Hierarchies

BVHs typically partition the primitives in a scene using a bottom up approach. A single primitive is enclosed within a bounding volume, typically a sphere or *axis-aligned bounding box* (AABB) for ease of intersection testing. The bounding volume can be expanded to include more primitives up to a predefined capacity. The process is repeated by enclosing the leaf nodes within larger bounding volumes, eventually resulting in a single root bounding volume.

Traversal of a BVH proceeds trivially. A ray is tested against the root node of the BVH tree. A successful intersection results in a recursive application of the intersection test to all children of the intersected volume. When a leaf node is intersected, testing takes place on the bound primitives.

Spatial Subdivision

Spatial subdivision takes an alternate approach by seamlessly partitioning the space in a scene, rather than the primitives. Two common examples of this top-down approach are the octree and uniform grid. Octrees recursively subdivide space into octants, each of which contains either empty space or primitives to be tested. An

initial bounding box is split along its axes and the primitives contained in that volume are tested for membership in against each octant. This process is applied recursively to any octant whose membership count exceeds a predefined threshold. Recursive application of ray/octant intersection tests ultimately determine which, if any, primitive was intersected [Foley et al. 1990].

Uniform grid representations non-recursively partition space into a 3D array of identical rectangular prisms. An initial bounding box is subdivided along each local axis according to the desired number of subdivisions, typically defined to achieve nearly cubic dimensions. Primitives are then tested against the subdivision volumes to determine membership. An intersection test against a uniform grid acceleration structure is accomplished by determining the entry point of the ray and iteratively traversing the subdivision volumes. This process is analogous to drawing a line across a discretized surface like a computer screen, only in 3 dimensions. When a populated volume is intersected, the primitives contained within are tested; a successful intersection test halts the traversal [Amanatides and Woo 1987].

4.4.2 Ray Intersection Testing

Ray intersection tests were performed against a bounding volume in the form of an axis aligned bounding box and a geometric primitive in the form of a triangle. There were slight variations between CPU and GPU implementations, but the algorithms used were identical.

Ray/AABB Intersection

AABBs were selected as the bounding volume of choice in part due to the efficient ray intersection test developed by Kay and Kajiya [1986]. A detailed listing of the algorithm is provided in Algorithm 4.1.

```
{define orthogonal bounding box using min/max of object}
BBmin ← [Xmin, Ymin, Zmin]
BBmax ← [Xmax, Ymax, Zmax]

{define ray as origin and direction}
Ro ← [Xo, Yo, Zo]
Rd ← [Xd, Yd, Zd]

{define near and far intersection distances}
Tnear ← −∞
Tfar ← ∞

{test the X pair of planes, Px}
if Rd.X = 0 then
  {R ∥ Px}
  if Ro.X < BBmin.X and Ro.X > BBmax.X then
    {no intersection}
  end if
else
  {calculate ray/plane intersections}
  T1 ← (BBmin.X − Ro.X)/Rd.X
  T2 ← (BBmax.X − Ro.X)/Rd.X

  if min(T1, T2) > Tnear then
    Tnear ← min(T1, T2)
  end if
  if max(T1, T2) < Tfar then
    Tfar ← max(T1, T2)
  end if

  if Tnear > Tfar then
    {no intersection}
  end if
  if Tfar < 0 then
    {no intersection}
  end if
end if

{test the Y pair of planes, Py}
{test the Z pair of planes, Pz}
{if all tests pass, an intersection has occurred}
```

Algorithm 4.1: Ray/AABB intersection testing using Kay-Kajiya slab algorithm

Ray/Triangle Intersection

Triangles have long been the primitive of choice when describing mesh geometries. Ray/triangle intersection testing has thus been a topic of intense study and development. The algorithm presented by Möller and Trumbore [1997] has been widely recognized as the fastest ray/triangle intersection test.

A detailed listing of the algorithm can be seen in Algorithm 4.2.

```
{define triangle as three points}
Tri ← [a, b, c]

{define ray as origin and direction}
Ro ← [Xo, Yo, Zo]
Rd ← [Xd, Yd, Zd]

{calculate u vector}
Edge2 ← Tri.c − Tri.a
P ← Rd × Edge2
T ← Ro − Tri.a
U ← T · P

{calculate v vector}
Edge1 ← Tri.b − Tri.a
Q ← T × Edge1
V ← Rd · Q

{if determinant is near zero, R lies in the plane of T}
det ← Edge1 · P
if (U > 0 and U < det) and ((U + V) < det) and (V > 0) then
  {intersection occurred}
  Dist. ← (Edge2 · Q) ∗ (1.0/det)
end if
```

Algorithm 4.2: Ray/Triangle intersection testing using modified Möller-Trumbore algorithm

4.4.3 Contrast Sensitivity Function

The hybrid technique requires that the scene be sampled in a perceptually valid manner. Of the many human visual system facets that can be exploited for this purpose, luminance was selected due to its direct relationship to mesh geometry

and its widespread use in perception literature.

Luminance is dependent on the orientation in space of a geometric primitive relative to both a light source and eye point. Since the hybrid technique seeks to down-select geometry, it is appropriate to discuss the rate of change in luminance between geometric primitives, referred to as contrast. Within a gaze-contingent context one can consider contrast at various displacements from the point of regard, typically expressed in units of visual angle. This implementation of the hybrid technique maintains compliance with the human visual system through examination of a contrast sensitivity function, which represents the perceptibility of the rate of change in luminance with respect to visual angle.

Contrast Sensitivity Function

Contrast sensitivity is determined through empirical measurements; there are several equations which match various data sets collected from human subject experiments. It is important to note that contrast sensitivity varies not only as a function of visual angle, but also as a function of velocity of the visual field. For the purpose of validating the hybrid technique the velocity component was held to zero, resulting in the most demanding human visual system requirements.

Creation of a perceptually based intermediate mesh, defining points through which rays are cast, is achieved using a discretized approximation of a contrast sensitivity function. This is accomplished using a rearrangement of the contrast threshold function explored by Geisler and Perry [1998], which has been shown to correlate well with existing data sets.

The derivation starts with the contrast threshold equation (4.1):

$$CT(f, e) = CT_0 \exp(\alpha f \frac{e + e_2}{e_2}), \tag{4.1}$$

where f is the spatial frequency in cycles per degree, e is the visual angle with

respect to the point of regard (eccentricity), CT_0 is the minimum contrast threshold, α is the spatial frequency decay constant, and e_2 is the half-resolution eccentricity. Geisler and Perry have determined empirical values for CT_0, α, and e_2.

By setting the left hand side of (4.1) to the maximum contrast possible, i.e., 1 (unity), and rearranging terms we obtain:

$$e = \frac{e_2}{\alpha f} \ln(\frac{1}{CT_0}) - e_2 \tag{4.2}$$

The resulting function provides the eccentricity at which a given spatial frequency f no longer contributes to perception in the human visual system (see Figure 4.1). In order to construct a spatial frequency map for a monitor's display it is necessary to compute e at coordinates (x, y). Rearranging terms again gives:

$$f(x, y) = \frac{e_2 \ln(1/CT_0)}{\alpha(e(x, y) + e_2)} \tag{4.3}$$

The resulting spatial frequency map is shown in Figure 4.2. This map guides the sampling rate of the intermediate mesh by dictating the minimum cycles per degree at a particular point on the screen.

4.4.4 Ray Generation

It is necessary to discretize Equation 4.3 in order to generate the intermediate mesh (Figure 4.3) used to guide ray casting. Attempting to sample the equation with minimal divergence results in irregular ray distribution and banding. In order to prevent this form of visual distraction, discretization takes place according to linearly increasing pixel spacing.

The ratio between pixel extent and physical extent of the display is determined, allowing the eye/screen distance to be mapped into pixel space. With this information a set of equations representing the spatial frequency required for casting a ray through every pixel (every two pixels, every three pixels, etc.) can be gen-

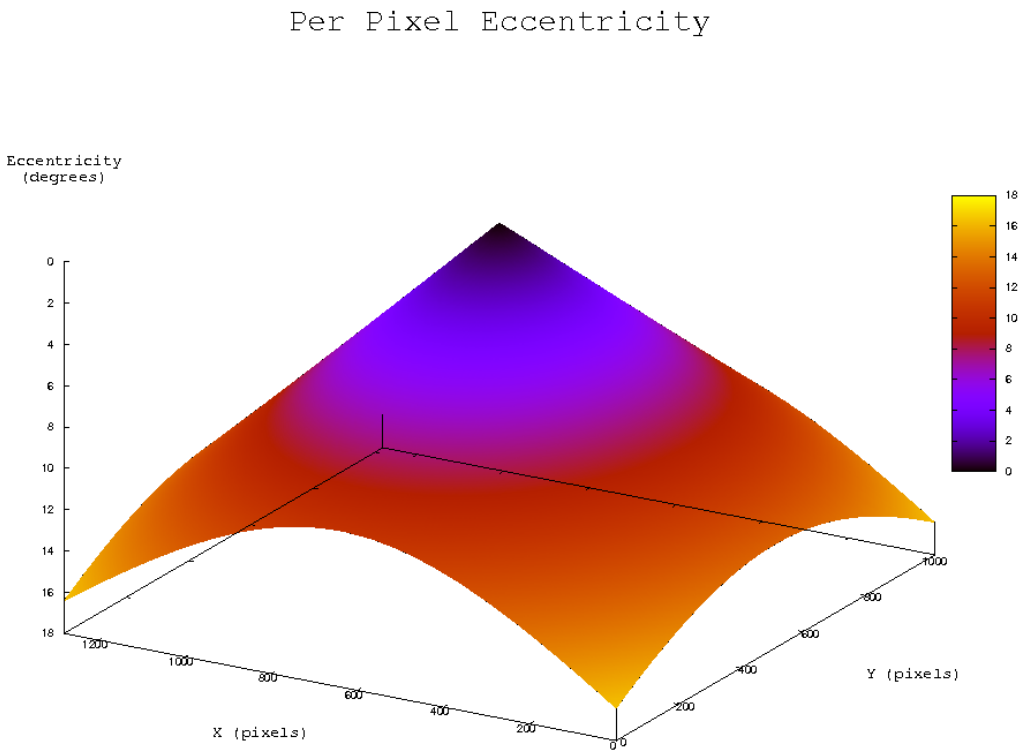

Figure 4.1: Per-pixel eccentricity with POR at center of 1280×1024 screen

erated. These equations, in units of pixels per degree (PPD), are obtained via: $PPD(n) = tan(1.0)/n, n \in [1..17]$

The first step in generating the intermediate mesh is the creation of a 1D array of pixel values representing rays to cast along the positive x-axis. The intersection points between this set of equations and Equation 4.3 indicate pixel spacing transitions in the 1D array. The 1D array is populated by inserting every integer from 0 to the first intersection point, then every second integer from the first intersection to the second intersection, and so forth until a value in the 1D array exceeds the maximum screen extent (see Figure 4.4).

Given the 1D array of pixel locations (corresponding to rays to be cast), generation of the 2D ray mask is accomplished by rotating the pixel values by some angular displacement. The displacement is determined by calculating the separation

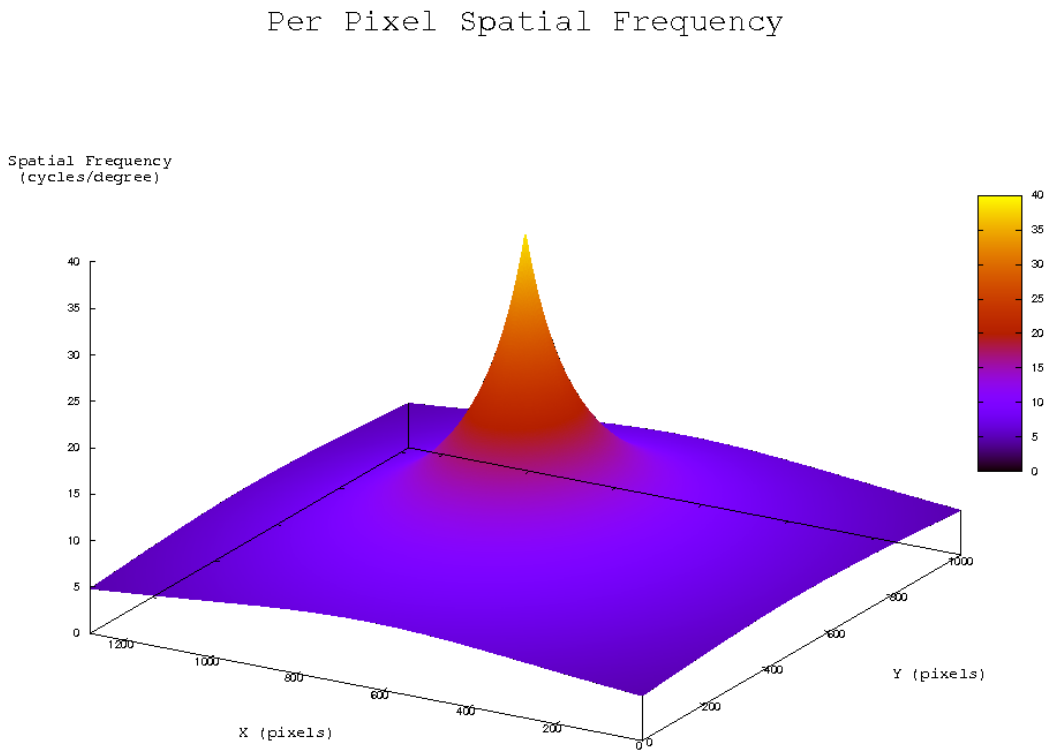

Figure 4.2: Spatial frequency of a 1280x1024 screen, with POR at center

of the outermost 2 entries in the 1D array and converting them from a pixel separation into an angular separation at the desired eye/screen distance. This angular displacement is used to rotate the points, whose positions are recorded in the 2D ray mask, until a full 360° is attained. Any pixels within the distance indicated in the 1D array's contiguous pixel region are filled in to account for discretization errors in the rotation process.

Finally, connectivity information is generated based on the ray mask. Each concentric circle of rays outside the contiguous region is joined with the succeeding circle of rays to form a quad strip. The combination of ray mask and connectivity information is rendered in place of the original scene geometry and is referred to as the intermediate mesh.

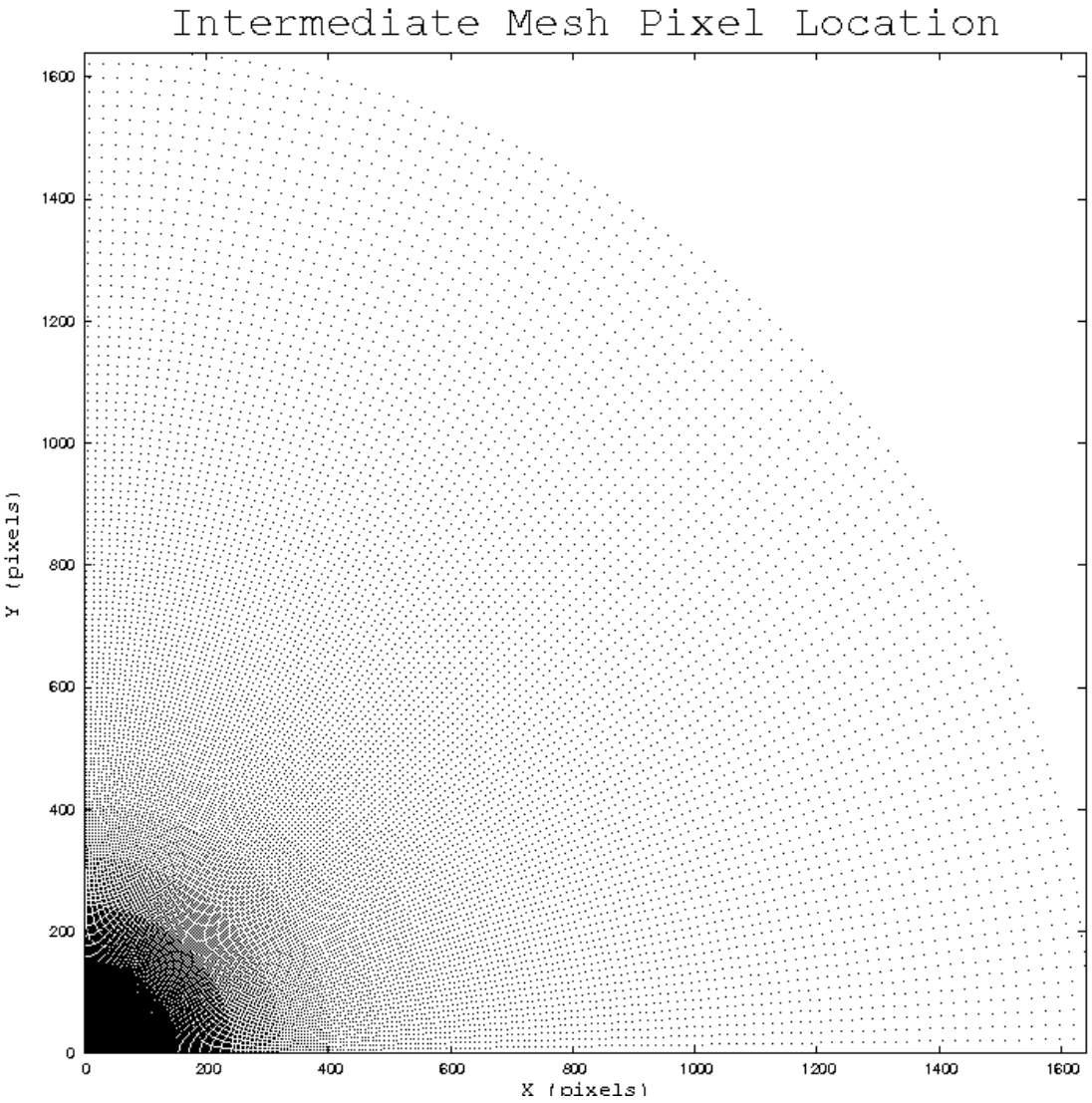

Figure 4.3: Intermediate mesh used for ray casting

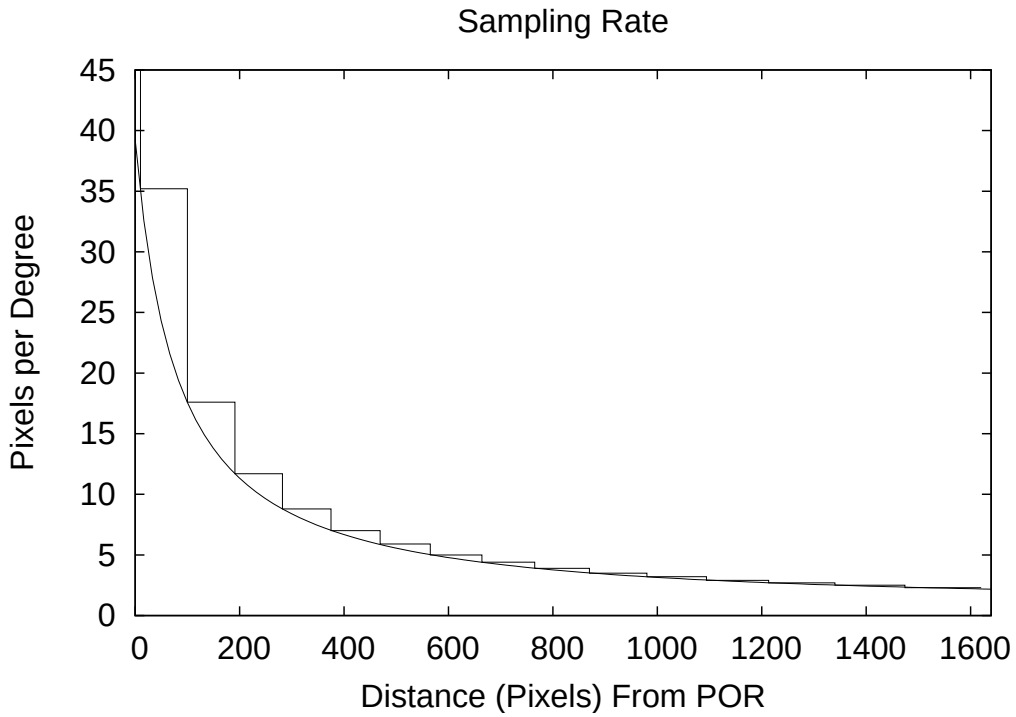

Figure 4.4: Plot of minimum sampling rate derived from CSF, with superimposed discretized version used to guide ray casting

Chapter 5

CPU-based Implementation

The initial implementation of the hybrid technique took place on a standard desktop CPU. It was hoped that the reduced amount of geometry queried during ray casting would allow a CPU-based implementation to perform at adequate frame rates. The process consisted of two steps: an initial preprocessing step in which the scene geometry was mapped to a ray casting acceleration structure, and runtime sampling of the scene to generate the intermediate mesh.

5.1 Preprocessing Component

Preprocessing was pursued whenever there was an apparent benefit to the frame rate. Two aspects of the display process were identified and submitted to preprocessing: construction of the ray casting acceleration structure, and creation of the ray casting mask used to direct scene sampling.

5.1.1 Ray Casting Acceleration Structure

Please see Section 4.4.1 for a brief high-level discussion of acceleration structures.

An octree ray casting acceleration structure was selected for use in the CPU-based implementation of the hybrid technique. This process does not affect the original mesh; the octree is used as a mapping between the original mesh geometry and a hierarchy of axis-aligned bounding boxes (see Figure 5.1). An octree was selected based on its ease of implementation and speed of search space culling. Both aspects are a result of the inherently recursive nature of an octree.

An octree is a spatial subdivision data structure that partitions a region along orthogonal axes into eight subregions, or octants. Each octant may be recursively partitioned in the same manner. The resulting tree structure ideally locates a par-

ticular item using a query string of length no greater than $log_8 N$, where N is the total number of items.

Octree Construction and Population

The initial state of an octree is that of a root bounding box whose corner vertices are combinations of the minimal and maximal extents along each axis as determined by the mesh vertices. The root node is then subdivided into eight child nodes by partitioning the bounding box into two equal halves along each axis. Triangles from the mesh are subjected to a triangle/AABB intersection test against the immediate children of the root node; a successful intersection indicates membership in that child. If a triangle intersects two or more neighboring bounding boxes, each intersected node includes the triangle in its population list.

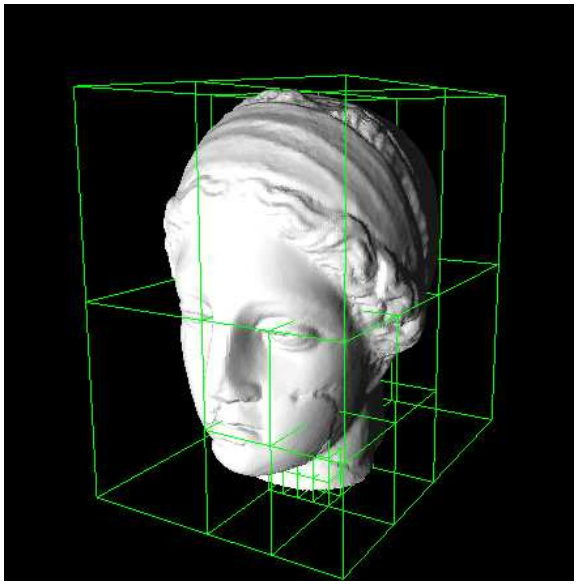

Figure 5.1: Octree construction

When the population of a node exceeds a predefined limit, the bounding box is subdivided into eight children through bisection along each axis. The triangles already assigned to the parent bounding box are redistributed into the eight children according to the triangle/AABB intersection test, leaving the parent bounding box

empty. Thus only leaf nodes in the octree contain triangles. The algorithm for the process is described in Algorithm 5.1.

```
use object's per-axis min and max to define root bounding box, BB_0

set bin_size to maximum triangles per bounding box
push object's triangles onto stack
while stack not empty do
  T ← pop()
  insert(T,BB_0)
end while

{attempts to insert triangle T into bounding box BB}
insert(T,BB):
if T intersects BB then
  if BB is a leaf node then
    if |BB| < bin_size then
      BB ← BB ∪ T
    else
      uniformly subdivide BB into octants
      push() contents of BB onto stack
    end if
  else
    ∀BB_n ∋ BB_n is a child of BB, insert(T,BB_n)
  end if
end if
```

Algorithm 5.1: Octree Construction

Separating Axis Theorem

Triangle/AABB intersection testing was accomplished efficiently through the use of Gottschalk et al.'s [1996] *Separating Axis Theorem* (SAT). The SAT can be stated as follows:

> **Theorem 5.1 Separating Axis Theorem**. Two polytopes are disjoint iff there exists a separating axis which is either perpendicular to a face of one of the polytopes or perpendicular to an edge from each. ∎

The intersection test is simple, and appears in Algorithm 5.2. All face normals and all cross products generated between an edge from both the triangle and the axis-

aligned bounding box are considered potential separating axes. Both polytopes are then projected onto the candidate axis; if no overlap occurs the axis is a separating axis and no intersection occurs.

5.1.2 Ray Generation

Please see Section 4.4.4 for a brief high-level discussion of the ray generation process.

5.2 Runtime Component

Extensive preprocessing reduced the required runtime actions to ray handling only. This included casting rays into the scene, assembling them into the intermediate mesh, and processing the mesh for the purpose of edge detection and refinement.

5.2.1 Ray Casting

The first step in the runtime process was casting the rays into the scene. This was broken into three parts: performing a ray/AABB intersection test against each octree in the scene to determine which meshes were intersected, traversing the octrees to determine which bounding boxes were intersected, then iteratively testing each enclosed triangle to determine if an intersection occurred. The rays themselves were easily defined given the ray mask from Section 4.4.4; the ray origin was set to the eye origin (set to the mouse location for testing purposes), and the ray direction was the difference between the appropriate location in the ray mask and the ray origin.

The recursive nature of octrees aided traversal when performing intersection testing. An initial test against the root axis-aligned bounding box determined if the ray could intersect the mesh. If an intersection was possible, the test was repeated against each of the children. A distance test against the resulting intersections determined the order in which each child was recursively tested.

If ray/AABB intersection testing encountered a leaf node, which by definition

```
BB ← AABB being tested
T ← triangle being tested
BB_T ← AABB defined by triangle's per-axis min/max
{consider normals as potential separating axes}
{test AABB normals against triangle}
if interval(BB.x) ∩ interval(BB_T.x) = ∅ then
  x-axis is a separating axis
end if
if interval(BB.y) ∩ interval(BB_T.y) = ∅ then
  y-axis is a separating axis
end if
if interval(BB.z) ∩ interval(BB_T.z) = ∅ then
  z-axis is a separating axis
end if

{test triangle normal against AABB}
determine near and far corners of BB wrt half-space defined by T
if (near ∈ half-space) ∧ (far ∈ half-space) then
  T⊥ is a separating axis
end if
if (near ∉ half-space) ∧ (far ∉ half-space) then
  T⊥ is a separating axis
end if

{consider all axes resulting from the cross product of every}
{edge in the triangle with every edge in the AABB as potential}
{separating axes}
for all E_T ∈ {T.a, T.b, T.c} do
  for all E_BB ∈ {BB.x, BB.y.BB.z} do
    {calculate potential separating axis}
    axis ← E_T × E_BB

    {determine interval of T projected onto axis}
    interval(T) ← min(E_T · axis),max(E_T · axis)

    {determine interval of BB projected onto axis}
    interval(BB) ← min(BB · axis),max(BB · axis)

    if interval(T) ∩ interval(BB) = ∅ then
      axis is a separating axis
    end if
  end for
end for
```

Algorithm 5.2: Triangle/AABB intersection testing using SAT

contained triangles, an exhaustive test of all triangles in that node was performed. The closest triangle intersection was maintained and reported when the list had been traversed.

5.2.2 Intermediate Mesh Construction

Subsequent to ray casting, the ray mask is treated as a mesh defined by points in space (from the original ray mask) and normals at those points (determined by the ray casting step). Although the points in the ray mask are defined in three dimensions, the Z dimension is constant, meaning that the intermediate mesh can be treated as a 2D object with 3D normal information. No data regarding connectivity of the points is saved during the ray generation step; rather, connectivity is determined mathematically.

The systematic approach taken to generating the ray mask lends itself well to the construction of the intermediate mesh. Every point P in each concentric ring R serves as a base index for a quadrilateral, which is defined by the points {(P,R),(P,R+1),(P+1,R+1),(P+1,R)}.

The region starting at the point of regard and extending to the limit at which pixels begin to be skipped in the discretized contrast sensitivity function is not a part of the intermediate mesh. Since a ray is cast through every pixel in that region, no benefit would be gained by constructing quads to approximate the unsampled regions.

5.2.3 Edge Detection

The human visual system is particularly sensitive to edge stimuli, both silhouette edges and internal edges [Hayward 1998]. In either case the edge can be viewed as a high rate of change in surface normals. Given that edges have a large perceptual impact on the human visual system, it becomes necessary to preserve edges as faithfully as possible. The edge detection step of the hybrid technique examines

the information returned in the intermediate mesh and uses it to guide additional ray casting to regions such as mesh boundaries. Figure 5.2 shows the silhouette of a mesh as it would appear if all geometry was rendered, with an overlay of the intermediate mesh showing the sampling rate in this region. If the intermediate mesh were rendered without the benefit of edge detection, the coarse edge reconstruction shown in Figure 5.3 would result. However, examining the rate of change of surface normals between vertices in the intermediate mesh allows mesh refinement along edges, and thus a more faithful representation as shown in Figure 5.4.

Edges are detected by iterating through all quads looking for discontinuities. For example, if any of the corners of a quad successfully intersected some piece of geometry, while another corner in the same quad did not, it is obvious that an edge exists between the points. It should be equally obvious that conditions can exist that would defeat this detection mechanism; however, these conditions would necessarily involve features with a spatial frequency beyond that being sampled, therefore presumably unimportant in terms of contrast sensitivity.

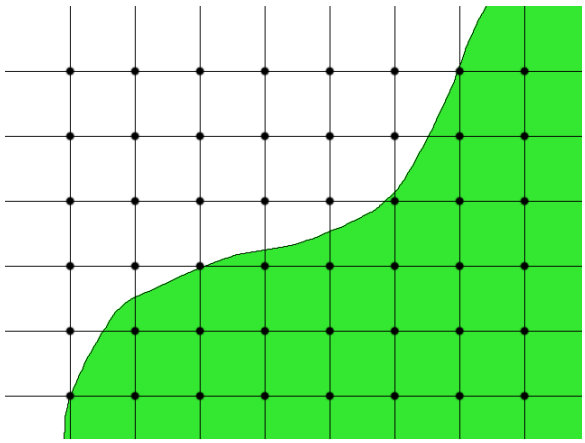

Figure 5.2: Intermediate mesh showing original mesh silhouette (shaded region)

Any quad which contains a silhouette edge is added to a list for refinement. Originally, refinement took the form of recursive subdivision of the quad; performance results indicated that it was faster to simply ray cast the entire quad. To do so, rays must be generated for each point in the quad, which is accomplished using

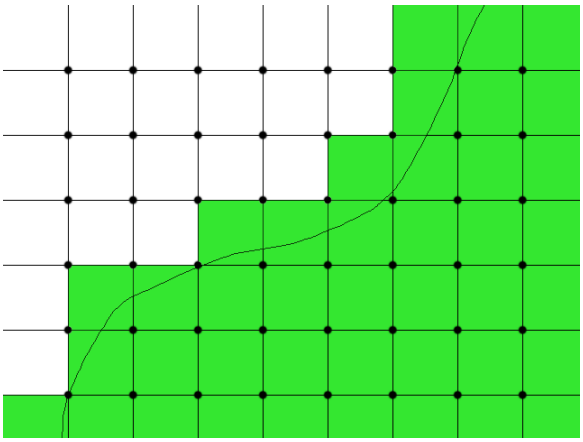

Figure 5.3: Intermediate mesh prior to silhouette preserving subdivision

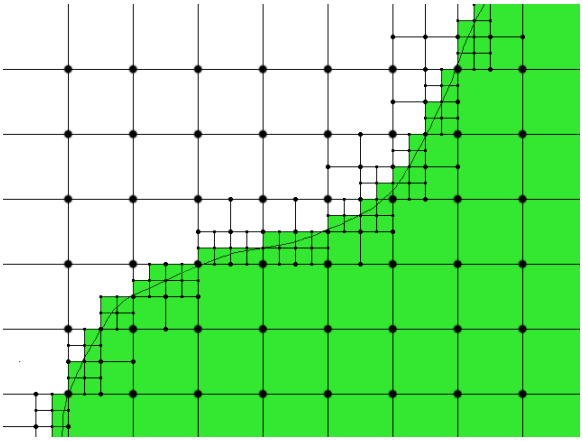

Figure 5.4: Intermediate mesh subsequent to subdivision

a modified Bresenham's [1965] algorithm to populate the edge lists. The resulting horizontal fragments are iterated across with each discrete point in the fragment corresponding to a new ray.

The final step is to render the resulting intermediate mesh. All single point data (such as edges and the foveal region) are also rendered.

5.3 Implementation Issues

Although the implementation was successful, the frame rate achieved was not. An informal pilot study indicated a maximum frame rate of 12 *frames per second* (FPS);

unacceptable for the planned search task. Furthermore, only five meshes were displayed at that frame rate, when at least nine meshes were desired for experimental purposes. The performance issues resulted in the decision to reimplement the hybrid technique using a GPU.

Chapter 6

GPU-based Implementation

Given that the CPU-based implementation of the hybrid technique provided insufficient frame rates for the desired testing regimen, alternative techniques were explored. Although the CPU-based implementation had been designed to scale well on multiprocessor computers, the trend towards GPU programming was too interesting to discard. GPU programming had previously shown a capacity for ray tracing at respectable speed; it was believed that exploiting the highly parallel nature of graphics cards for ray casting would permit interactive frame rates for the desired testing scenarios. While use of the manufacturers proprietary graphics language would undoubtedly offer superior performance, OpenGL was used to promote reproducibility on the widest range of GPUs possible.

The preprocessing and run time stages were significantly different from the implementation on the CPU. Different acceleration structures, and consequently different traversal algorithms, were selected for optimal performance on the GPU hardware. The details of the implementation are provided below.

6.1 Preprocessing Component

Four components used in the hybrid technique were precomputed in an effort to maximize GPU computational resources: the ray casting acceleration structure, the ray map, the intermediate mesh connectivity information, and the original mesh geometry. Both the ray map and connectivity information were derived from the contrast sensitivity function; the methods used to construct these data structures are discussed in Section 4.4.4. The acceleration structure used in the GPU implementation is significantly different from that used in the CPU implementation; a description is given below, along with reasons for its selection.

6.1.1 Ray Casting Acceleration Structure

Please see Section 4.4.1 for a brief high-level discussion of acceleration structures.

A uniform grid was used to accelerate ray casting. As with an octree, the mesh is not affected by the uniform grid, which exists as an overlay. The decision to use a uniform grid in this experiment was based on several factors. Uniform grids tend to perform well for scenes with uniform primitive density, such as the search task scenes used in this experiment [Wald et al. 2006]. In addition, the traversal algorithm for a uniform grid involves simple repetitive arithmetic, which allows a sequential representation in memory, as opposed to following links in a tree. The GPU used in the development of the hybrid technique was heavily penalized for texture look-ups, making link-based traversal, in which links can exist to widely separated memory locations, undesirable. Given that texture look-ups are inevitable, it was important to maintain the spatial locality of memory references to minimize texture cache misses. It was believed that storing a uniform grid by decomposing it into planes perpendicular to the major axis of the view vector (at successively greater distances from the eye location) would satisfy this requirement.

A uniform grid is a spatial subdivision data structure that uniformly partitions space along orthogonal axes. Several different philosophies regarding the proportions of the resulting rectangular solids have been proposed, but this implementation of the hybrid technique enforced approximately cubical proportions with the intention of distributing mesh data as equally as possible. This required an unequal number of subdivisions along each axis. In addition, the axes were selected to coincide with world axes, as was the case for the octree implementation.

Uniform Grid Construction and Population

The process used to create a uniform grid acceleration structure begins by determining a bounding box for the entire mesh. This serves as the initial cell in the uniform grid. Mesh primitives are placed in the appropriate cell in the bounding

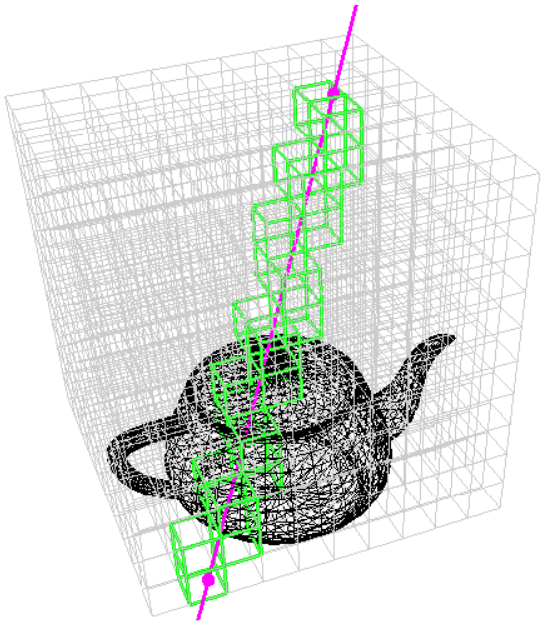

Figure 6.1: Example of uniform grid and traversal (from Christen [2005])

box until any cell exceeds an empirically derived binning size (40 triangles per cell worked well for the meshes used in this experiment). Primitives that span more than one cell are placed in all intersected cells. Triangle/AABB intersection tests used the Separating Axis Theorem as described in Section 5.1.1. Exceeding the binning size in any cell restarts the uniform grid construction process with an increased subdivision level. The process is repeated with the redefined uniform grid until all primitives are inserted without exceeding the binning size. Pertinent information, such as bounding box and cell dimensions, are recorded and stored with the uniform grid structure.

6.1.2 Storage in Texture Memory

The combination of strict adherence to OpenGL and the GPU used for initial development resulted in bulk memory access in the form of texture memory functions. Fortunately, the OpenGL 2.0 specification includes features such as non-power-of-2 rectangular textures and non-normalized floating point representations of the vari-

ous texture channels (red, green, blue, luminance, etc.) While these improvements over previous OpenGL texture definitions were helpful, interesting packing schemes were still necessary to organize the data structures in GPU memory.

Storage of the ray mask involved mapping the collection of points that guided ray casting, plus storage of the normal at that point, into a texture that contained four floating point pixels components: red, green, blue, and alpha value (RGBA). The intermediate mesh vertex positions were stored implicitly in the image; the X,Y screen positions of sample rays were considered the X,Y positions of the intermediate mesh points. The Z coordinate was set to zero, meaning that the screen coincided with the mesh. Storing the triangle normal at each point was easily accomplished through a 1 to 1 mapping between the {X,Y,Z} components of the normal vector and the {R,G,B} channels of a pixel. The alpha channel was used to indicate that the ray intersected scene geometry.

The uniform grid was stored in two textures. Each pixel in the first texture represented a cell in the uniform grid, defined as the number of primitives present in the cell and the initial offset into the list of primitives. These values were stored as a 2-tuple in the R and G channels of an RGB pixel. The offset referred to the second texture, which stored concatenated lists of offsets to primitives stored in the cell. As only a single value per offset was required, the texture used consisted of single luminance values only. The offsets in the second texture pointed to the original mesh geometry (see Figure 6.2).

The original mesh geometry was stored as a single list, with each triangle being defined by three points and a normal vector. Although there was a substantial penalty in terms of required storage memory due to high redundancy in point storage (each point was stored an average of 6 times), the cost of yet another memory redirection via texture look-up was deemed to be greater. Each point required three floating point numbers, as did the normal, both neatly fitting into consecutive RGB pixels.

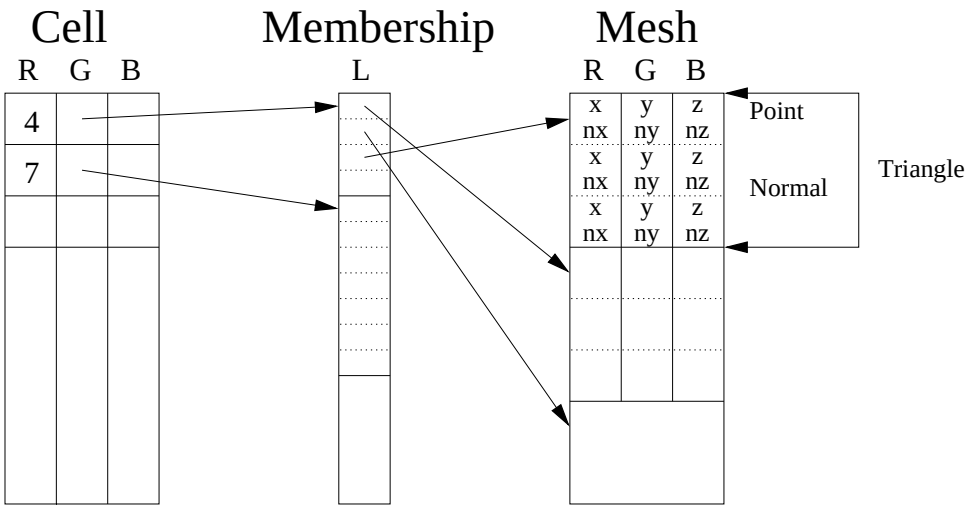

Figure 6.2: Storage of uniform grid in texture memory

6.2 Run time Component

The run time component of the GPU-based implementation consisted of casting the rays, copying the intermediate mesh back to CPU memory, examining the intermediate mesh for edges, drawing the quads defining the intermediate mesh, and providing increased sampling along edges. The GPU on which the method was developed did not provide geometry instancing capabilities, necessitating the CPU-based rendering and edge detection.

6.2.1 Ray Casting

Ray casting proceeded in a fashion similar to the process described in Section 5.2.1. Generating rays to sample the scene was accomplished by drawing a 2D filled rectangular bounding box on the screen around each of the meshes to be rendered. OpenGL rasterized the rectangle into fragments, which were passed to a fragment shader. A simple comparison with the pixels in the ray mask texture (offset by the point of regard) culled non-ray fragments. The remaining fragments, in conjunction with the ray mask, were used to generate rays. The eye position was defined as the center of the screen at approximately 54 centimeters from the screen. Fragments

surviving this stage progress through the uniform grid traversal shader.

Traversal of the uniform grid started with each fragment "color" initialized to 0.0; an intersection results in the alpha value being set to 1.0 and the RGB values set to the intersected triangle's normal. An initial intersection test against the uniform grid structure determines if the ray intersects anything; if it intersects the uniform grid, the initial cell is identified. Traversal is then accomplished by testing the cell for ray/triangle intersections and stepping forward along the ray to the next cell intersected (see Figure 6.1). If a triangle is intersected, it's normal is stored in the RGB portions of a pixel, and the alpha value is set to 1.0, indicating a successful intersection.

6.2.2 Intermediate Mesh Construction

The intermediate mesh was created by the CPU which necessitated a costly step; the entire framebuffer was copied to system memory. Then the ray mask was translated to the current eye positions X,Y coordinates, which was used to index the downloaded texture. This provided an X,Y,Z position for each point in the intermediate mesh, the normal at that point from the framebuffer download, and the intersection status of that ray (again, from the framebuffer download).

6.2.3 Edge Detection

The intersection data stored in the framebuffer is associated with quads in the intermediate mesh. All quads are then categorized by homogeneous and heterogeneous intersection types. A homogeneous quad contains uniform intersection data, all misses or all hits. A heterogeneous quad contains non-uniform intersection data. The original OpenGL render pipeline state is restored and the list of homogeneous quads is drawn with color information extracted from the copied framebuffer. A full ray casting shader is loaded and the heterogeneous list is drawn, with each rasterized fragment resulting in a cast ray. This departure from the adaptive refinement

approach taken in the CPU-based implementation was caused by the prohibitive cost of repeatedly copying video memory across the system bus. Performance tests indicated that simply ray casting through every pixel in an edge containing quad was more time efficient.

The result is a three layer image: the point of regard is rendered using the contiguous region of the ray mask, the homogeneous quads (internal mesh detail) rasterized on top of the initial ray casting results, and the heterogeneous quads (external silhouette edges) ray cast on top of both previous layers. The framebuffer is displayed and the process restarts.

6.3 Implementation Issues

In an effort to produce code executable on all major GPU designs, the shader programs used in this experiment were developed using the OpenGL Shading Language (GLSL). Although not all GPU specific capabilities were exposed, some of which may have been pertinent to accelerating the hybrid technique, GLSL proved satisfactory for the primary tasks of uniform grid traversal and ray/triangle intersection testing. There were, however, two issues worth noting.

Although implementation dependent, the most common upper bound on texture sizes in OpenGL is 4096×4096 pixels for any color representation (on modern hardware). While the resulting 16M floating point 4-tuples were capable of holding the mesh geometry, the uniform grid acceleration structure was more tightly constrained, consisting of an approximately 256×256×256 element design. This was sufficient, but reducing the geometry binning size would have resulted in an overly large structure. One possible solution would be to span the uniform grid acceleration structure over several textures; for example, eight 512×512×64 textures could be organized into a 512×512×512 element design.

Another issue affecting the uniform grid acceleration structure was the GPU's

use of an 8-bit loop counter and maximum of 65536 instructions in a shader program. Again, this constrained the maximum number of elements traversed to 256, as nesting the loop to give greater depth ran the risk of exceeding the maximum number of instructions available in a shader (loops were unrolled on the development GPU).

It seems likely that the suggestions in Section 8.2 may solve the aforementioned issues. These limitations would not be not present on the GPU used for experimentation, had the native API been used.

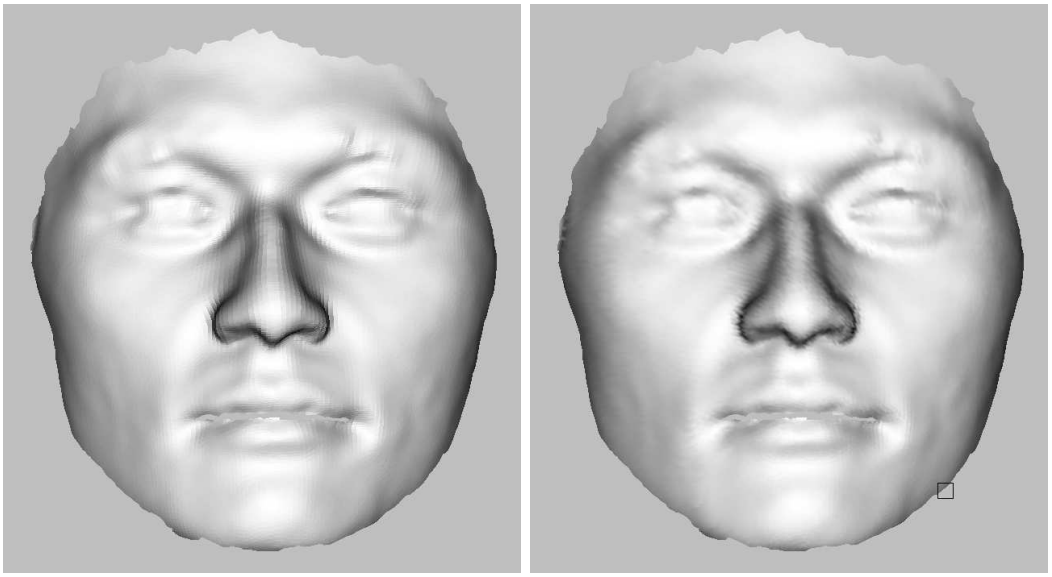

Figure 6.3: Original rendition (left), non-isotropically degraded rendition (right) with silhouette edges maintained and POR superimposed at lower right

Chapter 7

Human Subject Experiments

7.1 Methodology

It was hypothesized that varying the size of a high-resolution inset at the point of regard, while reducing peripheral detail in accordance with the contrast sensitivity function, would affect localization time in a visual search task. It was believed that maintaining silhouette edges would have a similar effect on localization time. Therefore, the objective of this experiment (null hypothesis) was to prove that no significant difference exists between the various factors under discussion.

7.1.1 Equipment

A Tobii ET-1750 video-based corneal reflection binocular eye tracker was used for real-time gaze coordinate measurement. The eye tracker operates at a sampling rate of 50 Hz with an accuracy typically better than 0.3° over a ±20° horizontal and vertical range using the pupil/corneal reflection difference [Tobii Technology AB 2003] (in practice, measurement error ranges roughly ± 10 pixels). The lower bound of the display program's refresh rate was informally measured at an average of 15 fps (the monitor's physical refresh rate is 30 Hz). The eye tracker's 17″ LCD monitor was set to 1280 × 1024 resolution. The eye tracking server ran on a dual 2.0 GHz AMD Opteron 246 PC (2 G RAM) running Windows XP. Although the Tobii allows limited head movement (30 × 15 × 20 cm volume), a chin-rest was used to maintain constant distance of approximately 54 cm from the monitor.

The client display application ran on a 2.2 GHz AMD Opteron 148 Sun Ultra 20 running the CentOS operating system.The client/server PCs were connected via the departmental 1 Gb Ethernet (both connected to a switch on the same sub-net).

A keyboard attached to the client system provided user input. The physical setup is shown in Figure 7.1 (the subject is shown interacting with the demonstration program).

The primary rendering engine consisted of an NVIDIA 8800GTX GPU. The 8800GTX has 768Mb of texture memory and 128 stream processors operating at 1.35 GHz. All ray casting occurred on-GPU; however, construction of the intermediate mesh necessitated CPU involvement.

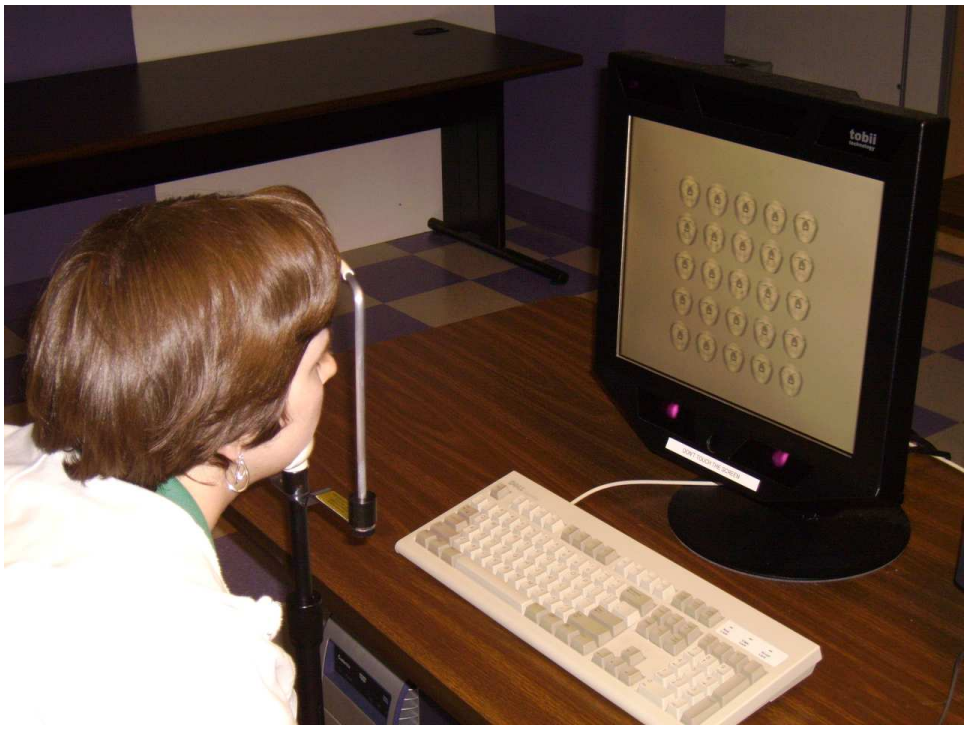

Figure 7.1: Example of equipment setup

7.1.2 Experimental Design

The independent variables used in this 2×5 repeated measures experiment were high-resolution inset size and the presence or absence of maintained object edges. Inset sizes spanned 2°, 5°, 10°, 15°, and 20° visual angle. For each inset size, object edges were either maintained or discarded, resulting in 10 combinations. Addition of a control (full screen ray casting) resulted in a total of 11 scene types to be tested.

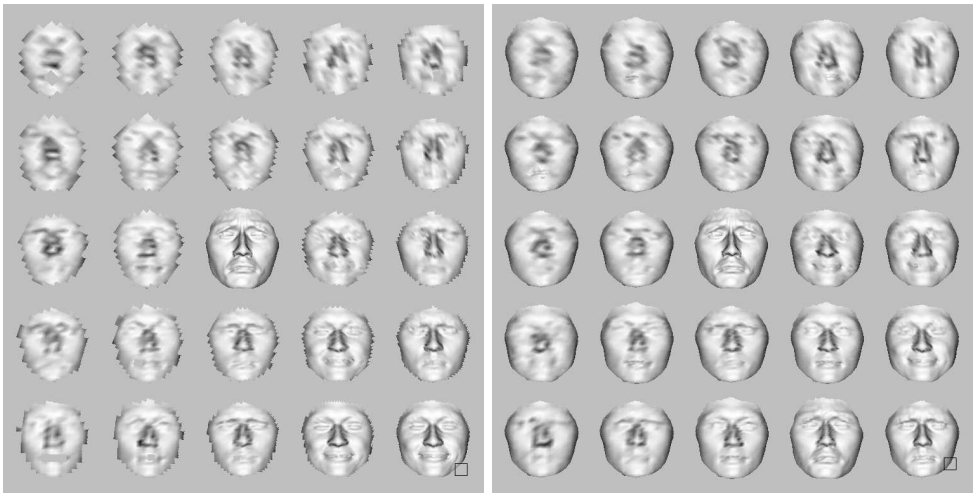

Figure 7.2: Sample search task (left), sample search task with silhouette edges preserved (right), each with box in lower right indicating POR

The dependent variable in this experiment was the visual search time required to localize a target. Search time started when the field of distractors was displayed and ended when the subject pressed the Space key.

The experiment utilized a within-subjects design, with each participant exposed to the aforementioned 11 scene types. A Latin square was used to control order effects. Each subject performed the 11 search tasks 4 times, resulting in 44 total trials per subject.

7.1.3 Human Subjects

There were 11 participants in the experiment, 4 female and 7 male. Subject ages ranged from 25 to 34, all with self-reported normal or corrected to normal vision. Although contact lenses were permitted, potential subjects wearing glasses were excluded from the experiment, as were others who failed the screening process (a total of 2) due to abnormally inconsistent eye tracking data.

7.1.4 Stimulus

A 9 point calibration was used at the beginning of a session, with a calibration accuracy confirmation scene displayed immediately thereafter. The calibration accuracy confirmation scene was also displayed at the mid-point of the experiment to assure that accuracy was within tolerances.

Four topologically consistent meshes displaying different facial expressions were used as targets. The original silhouette edges were consistent across all four meshes. Each target subtended approximately 3.5 degrees of visual angle at the screen distance used in the experiment. The entire scene subtended approximately 20 degrees of visual angle.

The scenes used for the visual search task consisted of a 5×5 grid of the meshes described above. The scenes were constructed by first selecting a target from the set of four meshes. Each mesh was selected at random from a pre-seeded pool, assuring that although the presentation order of each mesh was random, each mesh assumed the role of target the same number of times. The target mesh was then placed both in the center of the scene and at a random location in the grid. The remaining positions in the 5×5 grid were then filled at random with duplicates of the three non-target meshes.

7.1.5 Presentation

A scripted verbal explanation of the display program commands and execution flow was used to prepare the subjects for the demonstration program. The chair, chin rest, and keyboard were then adjusted for user comfort. At this point a demonstration program was run to reinforce the scripted explanation. The subjects were informed that the demonstration program functioned in exactly the same manner as the experimental program, with only the control group being displayed. The demonstration program was also used to introduce the meshes to be used in the experiment. Subjects were informed that time was not a factor in the demonstration,

and were encouraged to examine the different facial expressions in detail. Upon completion of the demonstration program the participants were permitted to ask questions and stretch.

Subjects started the experimental program when ready. A 9 point calibration was followed by a scene designed to allow the experimenter to judge the quality of the calibration. Recalibration was performed if the calibration was deemed to lack accuracy or precision. Following calibration, the target was displayed in the center of the screen. The subject was given as much time as needed to examine the target. The subject then triggered the search task, which displayed the 5×5 grid of distractors. Upon localizing the target within the grid, the participant pressed the Space key to finish the search task and load the next target. A break to check the calibration or stretch was allowed after 6 search tasks were completed. Upon completion of all 11 search tasks, users were encouraged to stand and stretch. The experimental program was run 4 consecutive times for each subject.

7.2 Results

7.2.1 Significance of Edges

One-way *analysis of variance* (ANOVA) indicates the effect of silhouette edges on time to completion is significant ($F(2,481) = 7.301$, $p < 0.01$). Pairwise comparisons using t-tests with pooled *standard deviation* (SD) suggest that visual search performance differs significantly between trials with edges maintained and trials with degraded (no) edges ($p < 0.01$; see Figure 7.3).

7.2.2 Significance of Window Size

One-way ANOVA indicates the effect of window size on time to completion is significant ($F(10,473) = 3.3805$, $p < 0.01$). Pair-wise t-tests with pooled SD indicate significantly slower search times with a 2° window with degraded silhouette edges

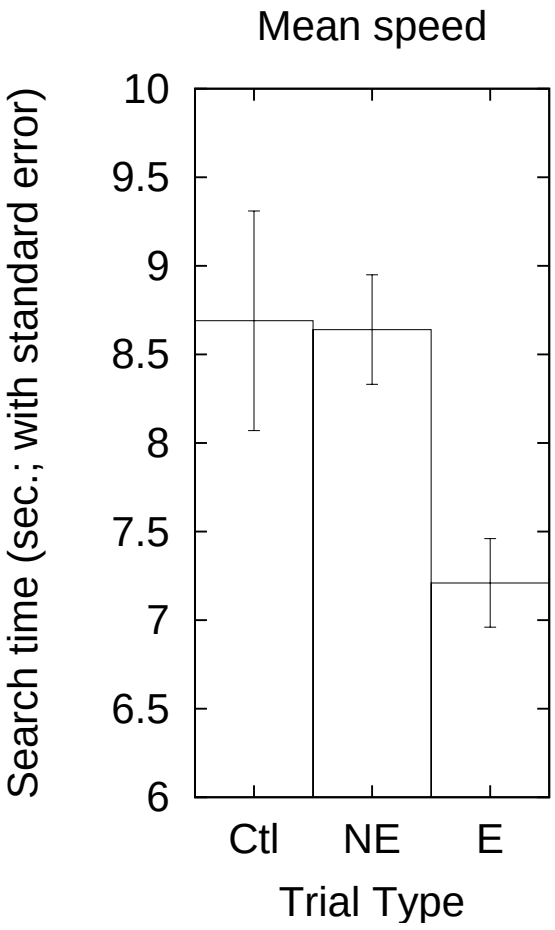

Figure 7.3: Mean Search Times (pooled; 'E' = silhouette edge preservation, 'NE' = no edges)

than compared to windows (with silhouette edges preserved) of 5° ($p < 0.05$) and 15° and 20° ($p < 0.01$; see Figure 7.4). A statistically significant difference was also expected between the edge-preserved window of 10° but none was observed.

7.2.3 Data

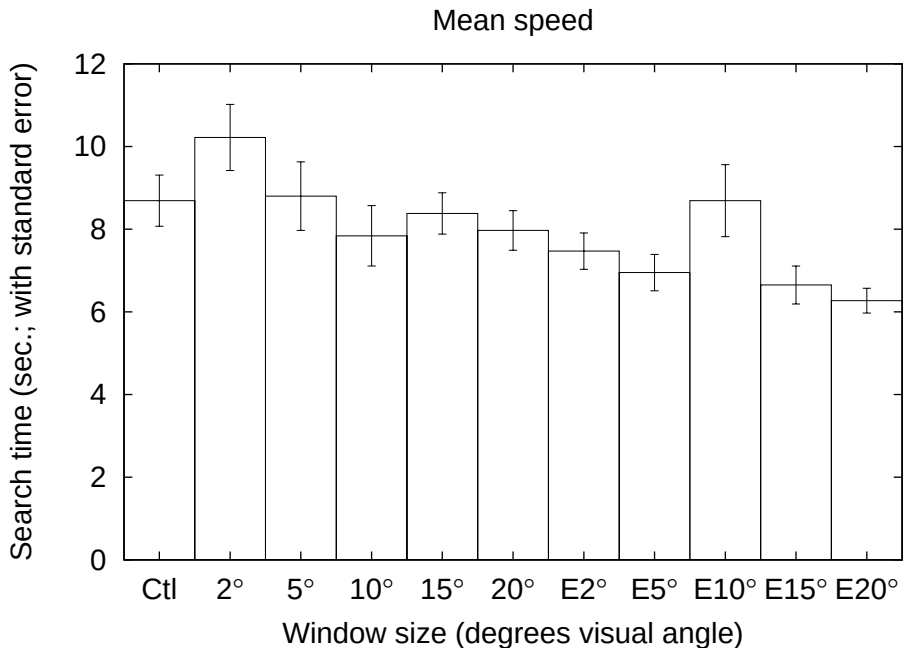

Figure 7.4: Mean Search Times (per window size condition; 'E' = silhouette edge preservation)

	2°	5°	10°	15°	20°	pooled
No Edges	10.22	8.80	7.84	8.38	7.97	8.64
Edges	7.47	6.95	8.69	6.65	6.27	7.21

Table 7.1: Mean Search Times (in sec.; Control = 8.69 s)

7.3 Discussion

As expected, search time decreased as window size increased and was lower still when silhouette edges were maintained. Search time was significantly prolonged with a 2° window with no silhouette edge information. Results suggest that performance is degraded with a very small foveal window when edge information is not preserved.

Given that the scene sampling rate was driven by an human visual system-based function, it is not surprising that the results of this experiment are similar to other

gaze-contingent rendering experiments. The trend shown in Figure 7.4, an inverse relationship between high-resolution inset and visual search time, is consistent with the image-based gaze-contingent rendering of Watson et al. [2004]. A visual search task using face images was performed using a $30^\circ \times 30^\circ$ high-resolution inset. Peripheral degradation was uniform within a scene (as opposed to the continuously varied peripheral degradation used in the hybrid technique), but varied between trials (see Chapter 3).

Our results are also similar to those obtained by Parkhurst and Niebur [2004], for a model-based visual search task. Their results are based on peripheral degradation generated through edge collapse, guided by normalized error in the collapse process. The visual search task utilized objects which varied widely in topology and profile (see Chapter 3).

In our case, a typical search started at the center of the screen, where the high resolution reference mesh was displayed, and swept out concentrically in search of the target. For example, as shown in Figure 7.5, targets are fixated in counter-clockwise order until the target is located in the upper left quadrant of the field of distractors. The subject then visually verified their selection against the reference mesh (note the multiple refixations). Fixations are detected via velocity-based analysis where a velocity $< 15^\circ/\text{s}$ denotes a fixation; the centroid of all congruent fixations determines the mean fixation coordinates.

There are two interesting observations depicted in Figure 7.4. First, the relatively long mean search time for the 10° window with edge preservation (E10) appears to be anomalous. A Latin square design was employed to reduce ordering effects, making it unlikely that a learning effect caused the increased search time. It should be noted that the standard error for E10 was the highest of any mean search time.

Second, the relatively poor performance of the Control group was unexpected. In the absence of any peripheral degradation, subjects should have been to able to localize targets rapidly. Anecdotal evidence, acquired through informal exit in-

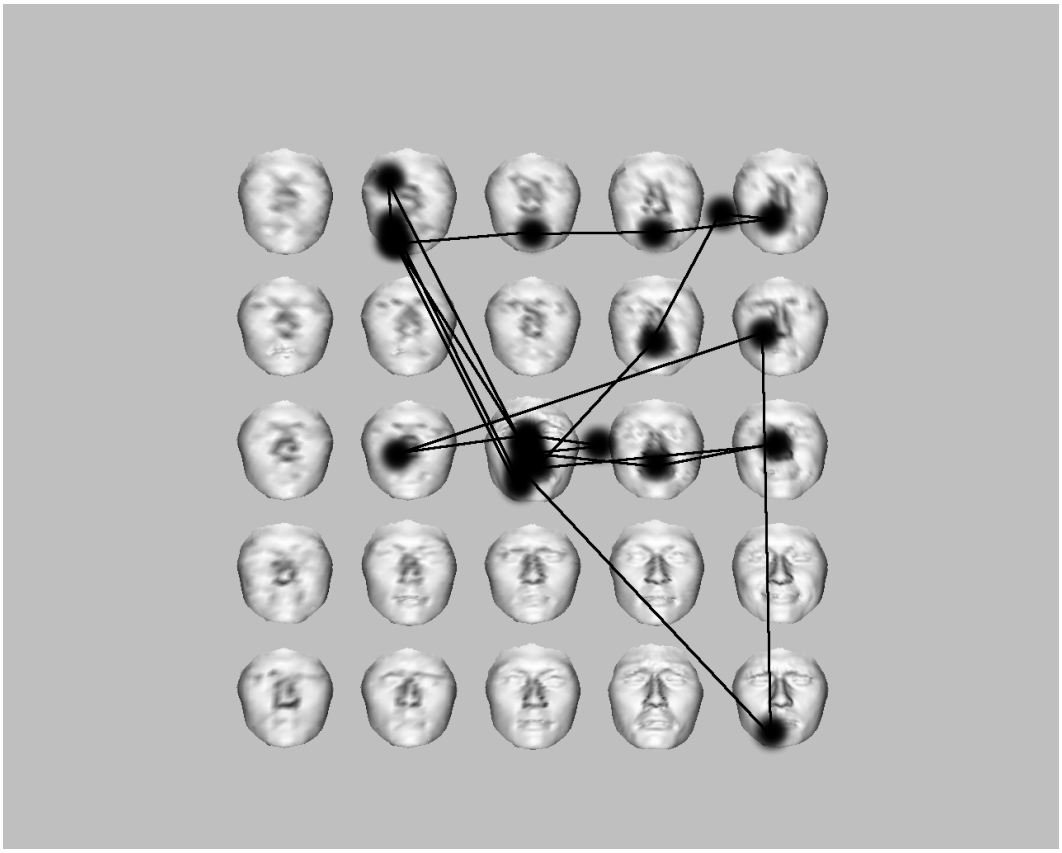

Figure 7.5: Example scanpath showing a subject's search strategy

terviews, indicates that the Control scenes were "overwhelming" and provided "too many possibilities." It appears that the lack of overt visual cues, coupled with the rich informational environment, precluded rapid localization of peripheral targets.

There is some support in the literature for the above interpretation. Specifically, Cave and Bichot [1999] suggest that for discrimination tasks, peripheral objects may increase reaction time if they distract the viewer during discrimination. Furthermore, the authors point out that the importance of attention in suppressing competing information from distractors is supported by neurophysiological studies showing that attentional modulation of neural responses is greatest when target and distractor both fall in the receptive field of a neuron, thus competing for representation by that neuron. By masking peripheral distractors, gaze-contingent rendering

may be beneficial to visual search where discrimination is a significant task component. The reader should be cautioned that this interpretation, although apparently in agreement with our observations, is rather tenuous. More controlled experimentation is required to substantiate this claim.

Chapter 8

Conclusion

The hybrid technique adds a method to the body of gaze-contingent rendering which brings together the two divergent research paths. Analysis such as edge detection is reduced from a complex task in three dimensions to a simple differencing operation in two dimensions. By non-isotropically sampling the scene space, the amount of geometry rendered is reduced, leading to the potential for render time savings. In addition to leveraging the best features of both image-based and model-based gaze-contingent rendering, the use of a highly decoupled sampling mechanism allows full exploitation of the massively parallel architectural design of modern GPUs.

8.1 Conclusions

A non-isotropic gaze-contingent rendering technique which utilizes aspects of both imaged-based and model-based rendering has been presented. The experimental results of the hybrid technique conform to prior results obtained using both image-based and model-based techniques. It would be instructive to duplicate the experiment using image-based and model-based degradation in order to directly compare the search times, but until the most recent generation of GPU, the geometry manipulation required of model-based techniques forced the algorithms to run on the CPU.

8.2 Future Work

Several possible mechanisms for accelerating or improving the hybrid technique became apparent by the end of the project. The built-in flexibility of the hybrid

technique will allow algorithmic enhancements, support research into alternative sampling methodologies, and due to adherence to OpenGL specifications, benefit from the increasing power of GPUs.

8.2.1 CSF Alternatives

Greater utilization of the human visual system could allow further reductions in the amount of geometry rendered. For the sake of presenting the most authoritative data regarding the hybrid technique, the contrast sensitivity function function used was intentionally forced into its most conservative form. It is possible to increase peripheral degradation either by setting Equation (4.1) to a non-maximal value of contrast, or by taking into account the velocity component of the contrast sensitivity function (see Daly et al. [2001]).

Additional human visual system components, such as chrominance or specular highlights, could be exploited to further decrease the sampling rate. Visual perception of certain colors diminishes as a function of visual angle, and could be used to reduce the number of peripheral rays. Specular highlights consist of regions of maximal luminance. Consequently, no detail is typically visible in these regions, and casting a large number of rays into them is wasteful. Integration of specular highlight detection could therefore benefit the hybrid technique.

8.2.2 Dynamic Ray Mask Generation

The ray masks used to guide scene sampling were computed for a specific eye distance from the screen. Failure to maintain that distance would lead to either oversampling the scene (too far away from screen), or undersampling which could lead to visually detectable artifacts. The use of a chin rest during the experiment was necessary to keep the subject's eyes at the proper distance from the screen and obtain valid data.

If the eye/screen separation could be determined on-the-fly, dynamically generated distance-correct ray masks could offer improved performance by sampling a

scene less frequently. Additionally, head movement would not result in a potential loss of visual fidelity through undersampling. The possibility of extracting a subject's distance from the screen directly from the Tobii equipment has not been explored, but seems plausible.

8.2.3 Reverse Ray Casting

A worst case scene for the hybrid technique involves geometric primitives with large screen coverage. The overhead of casting hundreds or thousands of rays through the same triangle would be less than desirable. Although meshes typically don't contain primitives that differ in area by orders of magnitude, they can subtend large expanses of screen area when very close to the viewer. If the vertices of inordinately large primitives were traced back to the eye position, the intermediate mesh could be stripped of rays to be cast into that region.

8.2.4 Intermediate Mesh Caching

Eye motions fall into two categories: fixations and saccades. A saccade is the ballistic change in position used to move the point of regard. A fixation is a relatively long period of time spent looking at a region. During fixations, the hybrid technique can repopulate the intermediate mesh with nearly identical data several times. Statistical data from multiple intermediate meshes could allow adaptive improvement of the resulting image through cumulative updating. Alternatively, a weighted average of several intermediate meshes could be used guide ray casting to regions with high rates of change, like edges, and away from low content regions.

8.2.5 Ray Casting Acceleration Structures

As mentioned in Chapter 4, a uniform grid spatial subdivision structure was used to accelerate ray casting intersection tests. The decision to use a uniform grid was partially based on the architectural restrictions specific to the GPU used during

development. Specifically, branching and texture look-ups were very costly. Use of the much more capable 8800GTX alleviates these concerns and should allow the use of more computationally efficient acceleration structures, such as a bounding volume hierarchy. While perhaps not ideally suited to the testing scenes presented during the experimental phase, "natural" scenes are more likely to benefit from BVHs.

8.2.6 On-chip Geometry Instantiation

Another GPU-based improvement is to exploit the on-chip geometry instantiation made available on the 8800GTX. Currently, the entire frame buffer is copied back into system memory after rays are cast. The image is accessed to generate the intermediate mesh, which is then sent back to the GPU to be rendered. It may be possible to create the intermediate mesh directly on the video card through the use of geometry shaders, simultaneously reducing traffic on the system bus and further exploiting the parallel nature of the GPU. This would also allow the possibility of adaptive subdivision of the intermediate mesh, reducing the number of rays cast to maintain silhouette edges.

APPENDICES

A Acronyms

AABB - Axis-Aligned Bounding Box

ANOVA - Analysis of Variance

BVH - Bounding Volume Hierarchy

CSF - Contrast Sensitivity Function

FPS - Frames Per Second

GCR - Gaze-Contingent Rendering

HVS - Human Visual System

LOD - Level of Detail

MRGM - Multiresolution Geometric Modeling

POR - Point Of Regard

ROI - Region Of Interest

SD - Standard Deviation

VE - Virtual Environment

B Forms Used in Experiment

Experimenter Script

1. This experiment involves a simple timed search task. Your goal will be to find a particular target within a field of potential targets as quickly as possible.

2. First you will be allowed to familiarize yourself with the eye tracker and the models used as stimuli. This will be accomplished using a demo program. Your goal in using the demo program is simply to become comfortable with the experimental setup in preparation for the search task. The actions you perform in the demo program will be identical to those in the experimental phase.

3. The first step in using the program is to calibrate the system. When you are comfortable, follow the on-screen prompt and press "r" to begin calibration. This is a standard 9 point calibration, starting in the top left portion of the screen and proceeding from left to right within a row, then top to bottom between rows. Your task at this point is simply to stare directly at the yellow target when it appears in each of the 9 successive points on the screen. Do not anticipate the next position and move your eyes prematurely; wait for the target to appear before looking at the next target. Please minimize blinking during the calibration phase.

4. When calibration has completed you will see 5 yellow targets with blue centers. Look at each of these targets at your leisure; they serve as confirmation that the calibration was successful. If the experimenter indicates that you need to recalibrate the system, you will receive individual instructions at that time to improve the calibration process. Follow the on-screen prompts to either recalibrate by pressing "r" or continue to the experimental phase by pressing "Space."

5. The experimental phase consists of two steps. The target you are searching for will be presented in the center of the screen. Familiarize yourself with the target. When you are ready to search for the target, press the "Space" bar to continue to the search task.

6. The search task leaves the target in the center of the screen, and surrounds it with a circle of potential targets to search. All targets in the circle are unique. Your task is to identify the target in the circle that matches the target in the center of the screen as quickly as possible. If necessary you may refer back to the target in the center of the screen. As soon as you are looking at the matching target in the circle, press the space bar. This will restart the search process and you will be presented with a new target to familiarize yourself with. This process is repeated.

7. During the search process you will be presented with the same 5 yellow targets with blue centers as was presented after the calibration phase. Again, look at each of the five targets at your leisure. This process assures that the calibration is still satisfactory. If the experimenter determines that recalibration is necessary, you will receive individual instructions on improving the calibration. Follow the on-screen prompts to either recalibrate by pressing "r" or continue to the experimental phase by pressing "Space."

8. The program will automatically exit when the experiment is finished.

Bibliography

Amanatides, J. and Woo, A. 1987. A fast voxel traversal algorithm for ray tracing. In *Eurographics '87*. Elsevier Science Publishers, Amsterdam, North-Holland, 3–10.

Arvo, J. and Kirk, D. 1989. A survey of ray tracing acceleration techniques. In *An introduction to ray tracing*. Academic Press Ltd., London, UK, 201–262.

Bailey, R., McNamara, A., Sudarsanam, N., and Grimm, C. 2007. Subtle Gaze Direction. In *SIGGRAPH '07: ACM SIGGRAPH 2007 Sketches*. ACM Press, New York, NY, USA, 44.

Bergström, P. 2003. Eye-movement Controlled Image Coding. Ph.D. thesis, Linköping University, Linköping, Sweden.

Bresenham, J. E. 1965. Algorithm for computer control of a digital plotter. *IBM Systems Journal 4,* 1, 25–30.

Cave, K. R. and Bichot, Narcisse, P. 1999. Visuospatial Attention: Beyond a Spotlight Model. *Psychonomic Bulletin & Review 6,* 2, 204–223.

Christen, M. 2005. Implementing ray tracing on gpu. M.S. thesis, University of Applied Sciences, Basel, Switzerland. URL: <http://gpurt.sourceforge.net/DA07_0405_Ray_Tracing_on_GPU-1.0.5.pdf> (last accessed October 2007).

Clarke, J. H. 1976. Hierarchical Geometric Models for Visible Surface Algorithms. *Communications of the ACM 19,* 10 (October), 547–554.

Daly, S., Matthews, K., and Ribas-Corbera, J. 2001. As Plain as the Noise on Your Face: Adaptive Video Compression Using Face Detection and Visual Eccentricity Models. *Journal of Electronic Imaging 10,* 1, 30–46.

Danforth, R., Duchowski, A., Geist, R., and McAliley, E. 2000. A Platform for Gaze-Contingent Virtual Environments. In *Smart Graphics (Papers from the 2000 AAAI Spring Symposium, Technical Report SS-00-04)*. AAAI, Menlo Park, CA, 66–70.

Dayal, A., Woolley, C., Watson, B., and Luebke, D. 2005. Adaptive frameless rendering. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Courses*. ACM Press, New York, NY, USA.

Duchowski, A. T. and Çöltekin, A. 2007. Foveated Gaze-Contingent Displays for Peripheral LOD Management, 3D Visualization, and Stereo Imaging. *Transactions on Multimedia Computing, Communications and Applications 3,* 4 (November). Accepted for Publication.

Duchowski, A. T., Cournia, N., and Murphy, H. 2004. Gaze-Contingent Displays: A Review. *CyberPsychology & Behavior 7,* 6, 621–634.

Eck, M., DeRose, T., Duchamp, T., Hoppe, H., Lounsbery, M., and Stuetzle, W. 1995. Multiresolution Analysis of Arbitrary Meshes. In *Computer Graphics (SIGGRAPH '95)*. ACM, New York, NY, 173–182.

Foley, J. D., van Dam, A., Feiner, S. K., and Hughes, J. F. 1990. *Computer Graphics: Principles and Practice*, 2nd ed. Addison-Wesley, Reading, MA.

Geisler, W. S. and Perry, J. S. 1998. Real-time foveated multiresolution system for low-bandwidth video communication. In *Human Vision and Electronic Imaging*. SPIE, Bellingham, WA.

Geisler, W. S., Perry, J. S., and Najemnik, J. 2006. Visual Search: The Role of Peripheral Information Measured Using Gaze-Contingent Displays. *Journal of Vision 6,* 9, 858–873.

Gottschalk, S., Lin, M. C., and Manocha, D. 1996. OBBTree: A Hierarchical Structure for Rapid Interference Detection. *Computer Graphics 30,* Annual Conference Series, 171–180.

Hayward, W. G. 1998. Effects of Outline Shape in Object Recognition. Vol. 24. Journal of Experimental Psychology, 427–440.

Hoffman, D. and Singh, M. 1997. Salience of Visual Partss. *Cognition 63*, 29–78. URL: <http://ruccs.rutgers.edu/∼manish/papers/hoffman_singh_1997.pdf> (last accessed June 2007).

Hoppe, H. 1997. View-Dependent Refinement of Progressive Meshes. In *Computer Graphics (SIGGRAPH '97)*. ACM, New York, NY.

Hunt, A. R. and Kingstone, A. 2003. Covert and overt voluntary attention: Linked or independent? *Cognitive Brain Research 18*, 102–105.

Itti, L. 2004. Automatic foveation for video compression using a neurobiological model of visual attention. *Image Processing, IEEE Transactions on 13,* 10, 1304–1318.

Jacob, R. 1995. Eye Tracking in Advanced Interface Desing. In *Advanced Interface Design and Virtual Environments*, W. Barfield and T. Furness, Eds. Oxford University Press, Oxford, England, 258–288.

Joyce, C., Gorodnitsky, I., King, J., and Kutas, M. 2002. Tracking Eye Fixations with Electroocular and Electroencephalographic Recordings. *Psychophysiology*, 607–618.

Kay, T. L. and Kajiya, J. T. 1986. Ray tracing complex scenes. In *SIGGRAPH '86: Proceedings of the 13th annual conference on Computer Graphics and Interactive Techniques*. ACM Press, New York, NY, USA, 269–278.

Levoy, M. and Whitaker, R. 1990. Gaze-Directed Volume Rendering. In *Computer Graphics (SIGGRAPH '90)*. Vol. 24. ACM, New York, NY, 217–223.

Lindstrom, P., Koller, D., Ribarsky, W., Hodges, L. F., Faust, N., and Turner, G. A. 1996. Real-Time, Continuous Level of Detail Rendering of Height Fields. In *Computer Graphics (SIGGRAPH '96)*. ACM, New York, NY, 109–118.

Luebke, D. and Erikson, C. 1997. View-Dependent Simplification Of Arbitrary Polygonal Environments. In *Computer Graphics (SIGGRAPH '97)*. ACM, New York, NY.

Luebke, D. and Hallen, B. 2001a. Perceptually Driven Interactive Rendering. In *University of Virginia Tech Report CS-2001-01*.

Luebke, D. and Hallen, B. 2001b. Perceptually Driven Simplification for Interactive rRendering. In *Proceedings of the 2001 Eurographics Workshop on Rendering*. AAAI, London, UK.

Luebke, D., Hallen, B., Newfield, D., and Watson, B. 2000. Perceptually Driven Simplification Using Gaze-Directed Rendering. Tech. Rep. CS-2000-04, University of Virginia.

MacCracken, R. and Joy, K. 1996. Free-From Deformations With Lattices of Arbitrary Topology. In *Computer Graphics (SIGGRAPH '96)*. ACM, New York, NY, 181–188.

Marmitt, G. and Duchowski, A. T. 2002. Modeling Visual Attention in VR: Measuring the Accuracy of Predicted Scanpaths. In *EuroGraphics (Short Presentations)*. EuroGraphics, Saarbrücken, Germany.

Möller, T. and Trumbore, B. 1997. Fast, minimum storage ray-triangle intersection. *J. Graph. Tools 2,* 1, 21–28.

Murphy, H. and Duchowski, A. T. 2001. Gaze-Contingent Level Of Detail. In *EuroGraphics (Short Presentations)*. EuroGraphics, Manchester, UK.

Ohshima, T., Yamamoto, H., and Tamura, H. 1996. Gaze-Directed Adaptive Rendering for Interacting with Virtual Space. In *Proceedings of VRAIS'96*. IEEE, 103–110.

Osterberg, G. 1935. Topography of the layer of rods and cones in the human retina. *Acta Ophthal. 6*.

O'Sullivan, C., Dingliana, J., Giang, T., and Kaiser, M. K. 2003. Evaluating the Visual Fidelity of Physically Based Animations. *Transactions on Graphics 22,* 3, 527–536.

O'Sullivan, C., Dingliana, J., and Howlett, S. 2002. Gaze-Contingent Algorithms for Interactive Graphics. In *The Mind's Eyes: Cognitive and Applied Aspects of Eye Movement Research*, J. Hyöna, R. Radach, and H. Deubel, Eds. Elsevier Science, Oxford, England.

Parkhurst, D., Culurciello, E., and Niebur, E. 2000. Evaluating Variable Resolution Displays with Visual Search: Task Performance and Eye Movements. In *Eye Tracking Research & Applications Symposium*. ACM, Palm Beach Gardens, FL, 105–109.

Parkhurst, D. J. and Niebur, E. 2002. Variable Resolution Displays: A Theoretical, Practical, and Behavioral Evaluation. *Human Factors 44,* 4, 611–629.

Parkhurst, D. J. and Niebur, E. 2004. A Feasibility Test for Perceptually Adaptive Level of Detail Rendering on Desktop Systems. In *Applied Perception, Graphics & Visualization (APGV) Symposium*. ACM, 49–56.

Reddy, M. 1997. Perceptually Modulated Level of Detail for Virtual Environments. Ph.D. thesis, U. of Edinburgh.

Reddy, M. 1998. Specification and Evaluation of Level of Detail Selection Criteria. *Virtual Reality: Research, Development and Application 3,* 2, 132–143.

Reingold, E. M., Loschky, L. C., McConkie, G. W., and Stampe, D. M. 2002. Gaze-Contingent Multi-Resolutional Displays: An Integrative Review. *Human Factors 45,* 2, 307–328.

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

Rusinkiewicz, S. and Levoy, M. 2001. Streaming qsplat: a viewer for networked visualization of large, dense models. In *I3D '01: Proceedings of the 2001 symposium on Interactive 3D Graphics*. ACM Press, New York, NY, USA, 63–68.

Schaufler, G. and Stürzlinger, W. 1995. Generating Multiple Levels of Detail for Polygonal Geometry Models. In *EuroGraphics Symposium on Virtual Environments*. EuroGraphics, 33–41.

Schmalstieg, D. and Schaufler, G. 1997. Smooth Levels of Detail. In *Proceedings of VRAIS'97*. IEEE, 12–19.

Sun, Y., Page, D., and Paik, J. 2002. Triangle Mesh-Based Edge Detection And Its Application To Surface Segmentation And Adaptive Surface Smoothing. *IEEE International Conference on Image Processing 3*, 825–828.

Tanriverdi, V. and Jacob, R. J. K. 2000. Interacting with Eye Movements in Virtual Environments. In *Human Factors in Computing Systems: CHI 2000 Conference Proceedings*. ACM Press, 265–272.

Tobii Technology AB. 2003. Tobii ET-17 Eye-tracker Product Description (v1.1). URL: <http://www.tobii.se/> (last accessed January 2007).

Vogels, R. and Biederman, I. 2002. Effects of Illumination Intensity and Direction on Object Coding in Macaque Inferior Temporal Cortex. *Cerebral Cortex 12*, 756–766.

Wald, I., Ize, T., Kensler, A., Knoll, A., and Parker, S. G. 2006. Ray Tracing Animated Scenes using Coherent Grid Traversal. *ACM Transactions on Graphics*, 485–493. (Proceedings of ACM SIGGRAPH 2006).

Watson, B., Walker, N., and Hodges, L. F. 1997. Managing Level of Detail through Head-Tracked Peripheral Degradation: A Model and Resulting Design Principles. In *Virtual Reality Software & Technology: Proceedings of the VRST'97.* ACM, 59–63.

Watson, B., Walker, N., and Hodges, L. F. 2004. Supra-Threshold Control of Peripheral LOD. *Transaction on Graphics (SIGGRAPH'04 Proceedings) 23,* 3, 750–759.

Watson, B., Walker, N., Hodges, L. F., and Worden, A. 1997. Managing Level of Detail through Peripheral Degradation: Effects on Search Performance with a Head-Mounted Display. *ACM Transactions on Computer-Human Interaction 4,* 4 (December), 323–346.

Williams, N., Luebke, D., Cohen, J. D., Kelley, M., and Schubert, B. 2003. Perceptually guided simplification of lit textured meshes. *Proceedings of the 2003 ACM SIGGRAPH Symposium on Interactive 3D Graphics*.

Zorin, D. and Schröder, P. 2000. *Course 23: Subdivision for Modeling and Animation.* ACM SIGGRAPH, New York, NY. URL: <http://www.mrl.nyu.edu/dzorin/sig00course/> (last accessed 12/30/00).

Zorin, D., Schröder, P., and Sweldens, W. 1997. Interactive Multiresolution Mesh Editing. In *Computer Graphics (SIGGRAPH '97)*. ACM, New York, NY.