A Review on Eye-Gaze Tracking, GCDs, and Implemented Foveated Imaging Systems Using Optimized Methods

Farzin Farhadi-Niaki

Department of Systems and Computer Engineering
Carleton University
Ottawa, Canada
ffniaki@sce.carleton.ca

Abstract — Eye-gaze tracking has been an important and interesting area of research for quite some time now. In day-to-day applications, eye-gaze tracking can be used as a computer interface for both industrial and nonindustrial applications, which require hands-free installations. It can also be used to help disabled people to use computers for communication and for environmental control. To track the eye gaze we have to deal with three principle problems: detecting the eye, tracking the eye and detecting the gaze of the eye on the screen where a user is looking at. Furthermore, developing a foveated imaging system, implemented on a general purpose computer, greatly reduces the transmission bandwidth of images. By taking advantage of this fact, it is possible to create an image that is almost perceptually indistinguishable from a constant resolution image, but requires substantially less information to code it. This is accomplished by degrading the resolution of the image so that it matches the space-variant degradation in the resolution of the human eye.

Gaze-Contingent Displays (GCDs) attempt to balance the amount of information displayed against the visual information processing capacity of the observer through real-time eye movement sensing. Based on the location of the observer’s focus of attention, GCD content can be “tuned” through several display processing means: Screen-based displays, model-based, and Attentive User Interfaces (AUs) managing object-level entities (e.g., windows, applications). This paper briefly reviews past and present display techniques as well as emerging graphics and eye tracking technology for Gaze-Contingent Display development, and finally the state-of-the-art methods to implement the foveated imaging system utilizing the hardware advancement.

Index Terms—Eye tracking, foveation, gaze-contingent displays, level-of-detail, wavelet.

I. INTRODUCTION

Gaze-contingent displays (GCDs) degrade the resolution of peripheral image regions, generally in a manner consistent with human vision, such as by degrading resolution matching human visual acuity. In such applications, an eye tracker is used to track the user’s gaze so that a foveal region moves with the user’s (overt) focus of attention. GCDs are important for both the study of human vision and eye movements, as well as for the reduction of computational effort or bandwidth in peripheral regions during image transmission, retrieval, or display.

Gaze-contingent displays have been invaluable for the purpose of studying visual perception. By removing information beyond perceptual limits, GCDs match the resolvability of human vision. In vision research, GCDs extend the classic “moving window” experimental paradigm [26] originally developed in reading studies. Current research efforts extend this classic but one-dimensional form of display to two and three dimensions through spatial degradation of images and spatio-temporal degradation of video.

In applied work, GCDs help increase display speed through compression of peripheral image information that is not resolvable by the user. Applications include flight and driving simulators, virtual reality, infrared and indirect vision, remote piloting, robotics and automation, teleoperation and telemedicine, image transmission and retrieval, and video teleconferencing [1]. In at least one instance GCDs inspired a nongaze-contingent approach to the application of a visual eccentricity model of the human visual system’s contrast sensitivity function, or CSF, to videophone compression based on face tracking [7].

In virtual environments three-dimensional (3D) stereoscopic displays are used, but are usually not gaze contingent.

Model-based manipulation of level-of-detail (LOD), where geometric models are reduced in detail at distance as well as eccentricity has also been explored in the last ten years.

While gaze-contingent techniques have evolved substantially over the last three decades, the image degradation methods employed by researchers have generally remained based in software. Perhaps due to the interdisciplinary makeup of this community, its members may not be fully aware of graphical approaches that are simple and elegant, yet exceptionally robust in terms of their capabilities for control of spatial as well as chromatic peripheral degradation. The latter is of particular importance, since peripheral chromatic degradation has not yet been fully explored, and any potential bandwidth savings have not yet been investigated.

Gaze-contingent displays are susceptible to two major sources of display lag. First, any eye tracker used in the enterprise will exhibit a delay in delivering its real-time gaze coordinates. State-of-the-art eye-tracking devices such as those from Tobii Technology [34], relying on video scenes of the face and eyes, generally incur a delay inversely proportional to
their sampling rates. This delay can range from 5–20 ms or more, depending on the type of camera used. Second, given the gaze coordinates (considered instantaneous), GCDs must incur an additional delay by reconstructing an image degraded by applying a degradation function to image regions peripheral to the center of gaze. Common software-based image decomposition approaches (e.g., pyramidial schemes such as Laplacian or wavelet) addressing this delay depend on preprocessing a given image, storing numerous degraded images in memory, and calling up the one with its foveally processed region closest to the instantaneous point of gaze.

While the delay incurred by eye tracking is one that cannot be obviated without effective use of eye movement prediction schemes, the image reconstruction bottleneck can for the most part be eliminated through application of hardware-assisted display techniques. The contribution of this article is a fragment shader for real-time gaze-contingent image reconstruction, performed on the graphics processing unit (GPU) of a modern PC graphics card. Although the shader only requires 5 lines of GLSL code, hardly warranting more than a page of documentation, the purpose of this work is to promote awareness of this elegant method in the eye-tracking community as well as the vision science and perceptual graphics communities, and to suggest the potential for spatiochromatic peripheral degradation research, which has not yet been attempted.

The article is composed in two major sections. First, a brief historical review of GCDs is given, focusing on the display techniques used previously. Second, the GPU-based image reconstruction program is developed through an evolutionary exposition of hardware-assisted methods, beginning with a multitextured approach and culminating with the simplest yet most robust (in terms of spatiochromatic rendering) GLSL program.

II. GAZE-CONTINGENT DISPLAYS: REVIEW AND CURRENT TRENDS

The following sections review the Ref. [10].

Based on the assumed knowledge of the instantaneous location of the observer’s focus of attention, GCD content can be “tuned” through several display processing means. Screen-based displays alter pixel level information generally matching the resolvability of the human retina in an effort to maximize bandwidth. Model-based displays alter geometric-level primitives along similar goals. Attentive User Interfaces (AUIs) manage object-level entities (e.g., windows, applications) depending on the assumed attentive state of the observer. The following sections briefly review past and present display techniques as well as emerging graphics and eye tracking technology for Gaze-Contingent Display development.

A. Background

Gaze-Contingent Displays (GCDs) degrade the resolution of peripheral image regions in order to reduce computational effort during image transmission, retrieval, or display. In gaze-contingent implementations, the high resolution region moves with the user’s focus of attention. An eye tracker is typically used to track the user’s gaze. GCDs help increase display speed through compression of peripheral image information, which is not resolvable by the user. Applications include flight and driving simulators, virtual reality, infrared and indirect vision, remote piloting, robotics and automation, teleoperation and telemedicine, image transmission and retrieval, and video teleconferencing [1].

By manipulating the display in real time, GCDs can provide compelling visualizations of visual field defects. GCDs can thus be used to educate students, physicians, and patients’ family members about the perceptual and performance consequences of vision loss [20].

GCD research has progressed from simple image-based stimuli (e.g., sine-wave gratings in perceptual research) to complex image-based stimuli (images and video), and more recently to model-based stimuli (e.g., 3D graphical models). Generalizing on this concept, Attentive User Interfaces, or AUIs, control arbitrary objects concomitantly with the user’s tracked attentional focal point.

B. Attentive User Interfaces

Attentive User Interfaces, or AUIs, are an instance of Non-Command Interfaces [22] where screen objects, or physical devices, are controlled by gaze. By monitoring users’ physical proximity, body orientation, and eye fixations, AUIs can be used to control physical objects such as light fixtures and television sets [32].
C. Model-Based Graphical Displays

In graphical systems, model-based methods aim at reducing resolution by directly manipulating graphical model geometry prior to rendering. Real-time, or gaze-contingent, model manipulation is gaining importance particularly for the benefit of display speedup in immersive displays (e.g., Virtual Reality, or VR) or complex graphical environments (e.g., composed of voluminous data such as millions of triangles).

Clarke [6]’s original criteria of using the projected area covered by the object for descending the object’s Level Of Detail (LOD) hierarchy is still widely used today. However, LOD management typically employed by these polygonal simplification schemes relies on pre-computed fine-to-coarse hierarchies of an object. This leads to uniform, or isotropic, object resolution degradation.

Recently, Parkhurst and Niebur [29] evaluated two perceptually adaptive rendering techniques, one velocity-dependent and one gaze-contingent. Decreasing gaze-contingent peripheral geometric detail was found to increase object detection reaction times. Reaction times to localize a target, however, decreased. This suggests that isotropic gaze-contingent LOD impedes target identification while the resultant increased frame rate facilitates virtual interaction.

Since acute resolvability of human vision is limited to the central 2–5°, object resolution need not be uniform. Due to the advancements of multiresolution modeling techniques, and to the increased affordability of eye trackers, it is feasible to extend the LOD approach to gaze-contingent displays, where models are rendered nonisotropically.

Another novel approach to gaze-contingent modeling for real-time graphics rendering was taken by O’Sullivan et al. [28], who considered temporal resolution in the periphery. More precisely, O’Sullivan et al. developed a degradable collision handling mechanism to limit object collision resolution outside the central display region. Highly prioritized object collisions in the central region are allocated more processing time so that the contact model and resulting visual response is more believable.

In the example shown in Fig. 3 (left), a reduction of the number of triangles by 70% still leads to an imperceptibly degraded display [25].

D. Focus Plus Context Screens

A related display variant to GCDs which are not necessarily gaze-contingent but share the foveal/peripheral demarcation are focus plus context screens. Focus plus context screens achieve the high-detail/low-detail effect by combining a large, wall-sized low-resolution display with an embedded high-resolution screen [2]. For tasks involving large maps or detailed chip designs, focus plus context screens were shown to allow users to work from 20 to 35% faster than when using displays with the same number of pixels, but in homogeneous resolution or with multiple views. For an interactive driving simulation, users’ error rates were only a third of those in a competing multiple-view setup [2].

Idelix, a company that specializes in developing a novel variant of a type of focus plus context screen, has produced Pliable Display Technology, or PDT. The PDT differs from bi-resolution focus plus context screens since instead of providing the traditional foveoperipheral resolution demarcation, the PDT preserves the periphery at the image’s original detail while magnifying the foveal region. An example of a PDT image is shown in Fig. 5.

Fig. 3. “Gaze-contingent spatial and temporal LOD modeling. As the viewer focuses outside the room at the left of the rendering (image at left, courtesy of David Luebke), scene objects located at the right side of the room are rendered using a lower level of spatial detail, indicated by larger triangles (overlaid). Collisions between L-shaped objects (image at right, courtesy of Carol O’Sullivan and John Dingliana) are calculated at a higher level of temporal detail if located within the user’s current focus of attention.”[10]

Fig. 4. “Focus plus context screens consist of a large lowresolution display with an embedded hi-resolution screen. The iconic illustration (bottom right) shows the location of the high-resolution focus screen. The callout shows the difference in resolutions between the focus and the context area. From [10].”[10]

Fig. 5. “Application of eye-slaved PDT lens: original runway image (left), with magnified region (right).”[10]
E. Screen-Based Displays

Resolution management can be made dynamic if (1) the user’s gaze can be measured (e.g., by an eye tracker), and (2) the central high resolution region can be made to move with the user’s focus of attention.

Today’s improvements in eye tracking and imaging and graphics hardware fuel gaze-contingent display research by allowing researchers to vary information along multiple dimensions, e.g., spatial, temporal, and color resolution. For screen-based VR rendering the work of Watson et al. [35] is particularly relevant. Watson et al. studied the effects of Level Of Detail (LOD) peripheral degradation on visual search performance. The authors suggested that visual spatial and chrominance complexity can be reduced by almost half without degrading performance. Specifically, Watson et al. report that LOD must support a task-dependent level of perceptibility. Below this level, LOD should increase when eccentricity is high or contrast is low, and all scales of LOD (fine or coarse) are equally important.

III. FOVEATED GAZE-CONTINGENT DISPLAYS FOR PERIPHERAL LOD MANAGEMENT, 3D VISUALIZATION, AND STEREO IMAGING

The following sections review the Ref. [9] and Ref. [23].

Advancements in graphics hardware have allowed development of hardware-accelerated imaging displays. This part of the report reviews techniques for real-time simulation of arbitrary visual fields over still images and video. The goal is to provide the vision sciences and perceptual graphics communities techniques for the investigation of fundamental processes of visual perception. Classic gaze-contingent displays used for these purposes are reviewed and for the first time a pixel shader is introduced for display of a high-resolution window over peripherally degraded stimulus. The pixel shader advances current state-of-the-art by allowing real-time processing of still or streamed images, obviating the need for preprocessed or stored images.

A. Early Research

Two experimental paradigms, the moving window and the foveal mask, were developed in the mid 1970’s to explore eye movements and human reading strategies. Since then, these paradigms have been adapted to other domains, such as vision research and computer graphics.

B. Perceptually Lossless GCDs

A GCD is said to be perceptually lossless for a specified viewing distance and (instantaneous) gaze direction if the reconstructed display and the original appear identical to human observers when viewed from the specified distance. Prior research of GCDs has mostly focused on the perceptual or performance effects of reducing the spatial frequency (i.e., cycles per degree or bits per pixel) of peripheral image regions. For real-time display, preprocessed images would be recalled from memory on a “just-in-time” basis, that is, usually in relation to the location of the user’s eye-tracked point of gaze.

C. Implementation of a Foveated Image Coding System for Image Bandwidth Reduction

This section reviews the Implementation of a Space-Variant Imaging technique, which has been done by Kortum and Geisler [23].

a) Algorithm

Initialization, calibration and calculation of the space-variant resolution grid take place prior to image display. However, because of the simplicity of the resolution grid structure (which is described in greater detail below), it can also be recalculated in real time, if desired.

b) Resolution Fall-off Calculations

The need for real time performance requires a resolution structure that results in high computational efficiency in algorithmic implementations. The square symmetric pattern, shown in Fig. 7, is one such configuration. Because the resolution structure is specified in Cartesian coordinates, and each of the SuperPixels is square, pixel locations can be represented with a set of corner coordinates. This allows implementation of operations such as scaling and translation to occur using only addition. Starting in the south-west corner of the north-east pixel in ring i (the pixel at location $x_i$, $y_i$), the size of a SuperPixel is calculated according to the formula,

$$W_i = \frac{W_0}{\sqrt{1 + \frac{x_i^2 + y_i^2}{\epsilon^2}}}$$

where $W_i$ is the size of the SuperPixels in ring i (in pixels), $W_0$ is the size of the central foveal SuperPixel (in pixels), $x_i$ and $y_i$ are the distances along a diagonal from the center of the screen (in degrees), and $\epsilon$ is the half-resolution constant, expressed in degrees. This function is based on available perceptual data and is also consistent with anatomical measurements in the human retina and visual cortex. Specifically, when $\epsilon$ is between 0.8 and 1.2 the SuperPixel size is approximately proportional to the human resolution limit at each eccentricity. Thus, if $W_0$ is less than or equal to the foveal resolution limit then the foveated image will be indistinguishable from the original image (with proper fixation).

a) Gaze Tracking

After the initial SuperPixel gray level assignment, the program enters a loop in which the position of the eye is measured and compared against the last measured eye position. Fig. 8, shows an example of how the portion of the ResolutionGrid that is displayed changes as fixation changes; here a subject begins with center fixation, and moves his eyes...
Fig. 6. "General flow diagram of the operation of the foveated imaging system."[23]

Fig. 7. "Foveated imaging system SuperPixel pattern arrangement, which is called the ResolutionGrid."[23]

towards the northeast corner of the display device. As he does this, the amount of his eye movements (in the x and y direction) are added to the current ResolutionGrid coordinates, causing the foveated region to offset the same amount. Notice how the SuperPixels increase in size as they become further from the point of fixation.

Calculating a ResolutionGrid that is twice (4 times the area) the size of the viewable area allows a computationally efficient offset method to track eye position and update the display without having to recompute the ResolutionGrid each time the eye moves. In Fig. 9, the dark outline square is the viewable screen, with the remaining portion being the expanded ResolutionGrid, or so-called virtual screen. Adding the eye movement offset (the amount the eye has moved since the previous measurement) to the current ResolutionGrid coordinates is essentially the same as moving the ResolutionGrid to a position that coincides with current fixation location, while keeping the viewable screen in a fixed location.

Fig. 8. "Changes in Resolution Grid with Fixation."[23]

Fig. 9. "Since the ResolutionGrid is twice the size (4 times the area) of the viewable screen, recomputation of the ResolutionGrid is unnecessary because all eye positions in the viewable screen can be accounted for with a simple offset of the ResolutionGrid."[23]
D. Space-Variant Imaging and Foveation

The following sections review the Ref. [9].

Geisler and Perry [20] proposed a method to generate completely arbitrary variable-resolution displays. Their display depends on pyramidal preprocessing of the images prior to display [19].

Geisler and Perry’s space-variant imaging software produces smooth, nearly artifact-free images at high frame rates, but is limited to manipulation of spatial resolution. In image compositing parlance, this switch-matte operation makes gaze-contingent rendering immediately obvious: Simply preserve high-resolution pixels only at matte locations with $\alpha = 1$ and map pixels at lower matte luminance levels to lower-resolution pixels (e.g., from a bank of preprocessed images).

Since pyramidal reconstruction schemes draw pixel data from multiple levels of resolution, pyramidal image synthesis provides a smoothly degraded, convincing visualization of the human visual system, termed foveation. A particularly popular pyramidal approach relies on image decomposition via the discrete wavelet transform (DWT) with selective coefficient scaling and decimation prior to reconstruction [5] [12]. Provided that appropriate wavelet filters can be found, reconstruction exactly matches linear mipmapping.

Given an $N \times N$ image, assuming that $N$ is a power of 2 with $n = \log_2 N$, the original image $f(x, y)$ is subsampled and smoothed into $n + 1$ subimages via

$$f_j \left( \frac{x}{M^j}, \frac{y}{M^j} \right) = \frac{1}{M^j} \sum_{k=0}^{M^j-1} \sum_{m=0}^{M^j-1} f(x+k, y+m), \quad 0 \leq j \leq n,$$

where $M$ is a smoothing filter of size $2^n$\(^{-j}\), and $j$ is the resolution level. Eq. (1) generates projections of the original image onto $n + 1$ scaled subspaces equivalent to the subspaces generated by the scaling function of the DWT. The subspaces in this instance are scaled analogously to the DWT with resolution level $j = 0$ corresponding to the coarsest resolution level. Eq. (1) is a slightly different representation from the classical recursive pyramid approach, since each subspace is subsampled directly from the original image $f^n$, not from the image at the next-finer resolution level $f^{n+1}$. The wavelet pyramid is formed by the union of the original image and the set of subsampled images. Reconstruction of the image at a given pixel location $(x, y)$ depends on the desired resolution of the pixel.

The desired resolution level is bandlimited to the number of decomposed resolution levels (typically the decomposition is dyadic in nature) bounded by the two closest resolution subimages $f^{j-1}$ and $f^j$. The final pixel value at location $(x, y)$ is calculated as a linear combination of pixel intensities in the pyramid.

$$f(x, y) = (1 - p)f^{j-1} \left( \frac{x}{2^j}, \frac{y}{2^j} \right) + pf^j \left( \frac{x}{2^j}, \frac{y}{2^j} \right)$$

Eq. (2) represents linear intermap interpolation. To generate wavelet multiscale representations of a given image matching mipmap decomposition with the normalized box filter, the low-pass wavelet filter $\{h_k\}$ is set to $\{1/2, 1/2\}$. The detail filter is then the quadrature mirror of $\{h_k\}$, namely, $\{g_k\} = \{1/2, -1/2\}$. To guarantee perfect reconstruction, dual filters (signed of $h_k$ and $g_k$) are required to satisfy biorthogonal conditions. For filters of length 2, the following equations must hold.

$$h_0 + h_1 = 1, \quad g_0 + g_1 = 0$$

Dual detail filter coefficients are unnormalized versions of the Haar filters; that is, they are semi-orthogonal Haar wavelets (or prewavelets). Normalized Haar filters will generate the same reconstruction as mipmapping at dyadic-resolution boundaries, but will lose luminance information between boundaries where linear interpolation is required. The benefit of the semi-orthogonal wavelets is that correct luminance values will be generated at any desired resolution level. Note that the coefficients of the low-pass filter $\{h_k\}$ match the aforementioned averaging box filter exactly. This can be easily verified by obtaining the tensor product of the scaling filter at any decomposition level. For example, at the first level $(j = n - 1)$, the effective sampling filter is a $2 \times 2$ filter with cells equal to 1/4. At level $j = n - 2$, the filter is a $4 \times 4$ filter with cells equal to 1/16. Note that under the DWT, the finest resolution level (i.e., $j = n$, the original image) is not present in the pyramidal transformation.
To obtain interpolation results identical to mipmapping, an intuitive approach would be to maintain reconstructed scaled subimages produced by successive steps of the inverse DWT, and then to perform the interpolation step between the subimages. Assuming that equivalent subsampling filters guaranteeing perfect reconstruction are used, for example, orthogonal filters, this approach yields identical results to mipmapping, although it is memory intensive. What is perhaps not obvious is that identical interpolation results can be obtained by scaling wavelet coefficients prior to reconstruction. Wavelet coefficient scaling results in attenuation of the signal with respect to the average (low-pass) signal. Full decimation of the coefficients (scaling by 0) results in a lossy, subsampled reproduction of the original. Conversely, scaling wavelet coefficients by 1 preserves all detail information, producing lossless reconstruction. Selectively scaling the coefficients by a value in the range [0, 1], at appropriate levels of the wavelet pyramid, produces a variable-resolution image upon reconstruction. This approach is equivalent to mipmapping reconstruction with linear interpolation of pixel values.

In mipmapping, the value of interpolant \( p \) is determined by an arbitrary mapping function which specifies the desired resolution level \( l \). The two closest pyramid resolution levels are then determined by rounding down and up to find subimage levels \( j - 1 \) and \( j \).

To scale wavelet coefficients, \( p \) is set to 0, 1, or the interpolant value at particular subbands according to the following relation dependent on \( l \):

\[
P = \begin{cases} 
1, & j < [l] \\
1 - [l], & j = [l] \\
0, & j > [l] 
\end{cases} 
\]

(4)

For example, if at some pixel location \((x, y)\), \( l = 1.5 \), then wavelet coefficients would be preserved (scaled by 1) at levels \( j \leq 1 \), scaled by 0.5 at level \( j = 2 \), and decimated (scaled by 0) at levels \( j > 2 \) at the appropriate pixel location in the subimages.

Given an \( n \)-length discrete 1D function \( f_j \) at resolution-level \( j \), its DWT decomposition is given by the relations

\[
f^{j-k}_{-}(x) = \sum_{h \in \mathbb{Z}} h_k f_{j-1}^{k}(2x + k), \quad f^{j-k}_{+}(x) = \sum_{g \in \mathbb{Z}} g_k f_{j-1}^{k}(2x + k), \quad h, k \in \mathbb{Z},
\]

where \( \{h_k\} \) and \( \{g_k\} \) are one-dimensional low- and high-pass filters, corresponding to the scaling and wavelet functions \( \varphi \) and \( \psi \), respectively. Reconstruction with coefficient scaling is then written as

\[
f^{j}_{-}(2x - t) = \sum_{k \in \mathbb{Z}} h_{-t, 2k} f^{j-k}_{+}(x) + p \sum_{k \in \mathbb{Z}} g_{-t, 2k} f^{j-k}_{-}(x) + p, \quad t \in \mathbb{Z},
\]

where \( t \) is the filter length, and \( p \) is the resolution-level dependent interpolant defined by (4).

The aforesaid relations extend directly to two dimensions through 2D tensor product assembly of the 1D filters. It can be shown that wavelet coefficient scaling is equivalent to linear pixel interpolation under mipmapping [11]. The proof is intuitive, since subimages in the mipmap pyramid correspond to the low-pass subimages recovered at each stage of the inverse DWT reconstruction.

Duchowski [12] provides an acuity-matching mapping modeling the human visual system derived from empirical MAR (minimum angle of resolution) data [17]. MAR data at the border of the projected foveal ROI (at 5° visual angle) is converted to expected maximum resolution in dots per inch (dpi). Expected resolutions at peripheral eccentricities are derived relative to this maximum. Depending on the viewing distance and resolution of the display device, relative resolvability values in dots per inch are then converted back to pixel units to give the diameters of resolution bands. Wavelet space coefficient degradation and a resultant image with 2 foveated ROIs are shown in Fig. 12.

E. Stereoscopic Imaging

Recently, Çöltekin [4] used a pyramid-based LOD management method for close-range stereo photogrammetric rendering. Even though all GCDs are not 3D and all displays do not use stereoscopic principles, stereoscopy is an obvious and relatively mature technology to utilize for a number of purposes. Recently, mobile devices such as a mobile phone and a laptop computer were produced by Sharp, Inc., with optional auto-stereoscopic displays and successfully marketed. Stereoscopic displays may offer certain other advantages as well. These include potential for better relative depth judgment, spatial localization, camouflage breaking, surface material perception, and better judgment of surface curvature [21]. It is also thought that stereoscopic vision improves visual acuity when compared to monoscopic viewing [3] [8].

F. Evolution of the GPU-Based GCD

Duchowski [13] refers to gaze-contingent display processing as either screen based or model based where the former depends on image processing and the latter on processing graphics primitives (e.g., triangles). Texture-mapping (and shader programming) for gaze-contingent image display is a hybrid approach. Peripheral degradation of the image still relies on image processing, albeit the image is now considered a texture map. Rendering of the image relies on mapping the image onto a simple graphical object, in most cases a polygon (usually a screen-aligned quadrilateral) of the same dimension as the display window.

There are several tradeoffs between the texture-mapping and screen-based approaches, although both are now typically provided by graphics libraries such as OpenGL [33].

Advantages of the screen-based approach include the following:

1) Image resolution is of minor importance. Provided that the viewing window is made to be the same size
as the given image, the resultant display is generally shown at 1:1 pixel mapping, namely, the image is drawn to scale.

2) Provided a graphics card that supports OpenGL’s imaging subset in hardware is used, image processing operations can be performed quickly via hardware-accelerated convolution.

3) Various blending operations are provided that enable simple image combinations to take place via an image’s alpha channel.

There are, however, disadvantages to the screen-based approach:

1) Not all graphics cards support (or supported) the imaging subset in hardware. Lacking hardware support for the imaging subset, imaging operations such as convolution reverted to software implementation. This resulted in noticeable speed degradation.

2) The most significant drawback of the screen-based approach for gaze-contingent display is that the required image-blending functions (for blending foveal and peripheral image portions) rely on the images’ alpha channels. Thus, to provide a GCD, the image alpha channels would need to be translated in real time to match the foveal region, a potentially prohibitively expensive operation.

Texture-mapping, and in particular multitexturing and related fragment programming, solves the blending problem, since the alpha channel can be dissociated from either foveal or peripheral image and made into its own image. This is an important point since once so dissociated, the alpha mask can be manipulated independently. The manipulation that is most relevant to gaze-contingent display is translation of the foveal mask. Since mask translation is performed quickly in hardware, the result is real-time movement of the foveal region.

To summarize the distinction between screen-based and texture-based approaches, texture-mapping offers much greater flexibility in image display at the expense of additional complexity.

G. Mipmapping

For fully hardware-accelerated display as discussed here, GCDs can utilize in-hardware image degradation provided by built-in mipmapping functions. Mipmapping provides a method of prefiltering an image (texture) at multiple levels-of-detail [35]. Mipmaps are dyadically (by powers of two) reduced versions of a high-resolution image. One can either create these images manually a priori, or have them created automatically by OpenGL.

Several filter options are available for generating coarsely subsampled or linearly interpolated images. Four texture minification options control combinations of inter- and intramap pixel interpolation. The effect of these commands generates a coarsely or smoothly degraded periphery for dyadic levels of degradation, exemplified in Table I for various levels of degradation. Next, two recent approaches based on mipmapping are briefly reviewed for completeness and comparison to the subsequent newly introduced fragment programming technique. The former approach is suitable for implementations on third-generation graphics cards, while the latter requires fourth generation cards (fourth-generation graphics cards are distinguished from earlier versions by their ability to compile and run so-called fragment as well as vertex shader programs). In Table I, the fovea is over the airplane in the upper-right quadrant. The top image row shows nearest-neighbor interpolation: a faster means of image reconstruction from multiple subimages at the cost of visible blocky artifacts. The bottom image row shows linearly interpolated reconstruction, leading to smoothly reconstructed images.

H. Multitexturing

Real-time rendering of a biresolution GCD relies on two images. The first requirement is the source image for generating a high-resolution inset as well as a low-resolution background. The low-resolution background image is generated by dyadically degrading in hardware the source image via OpenGL’s mipmapping facilities. Alternately, the source image (or another image altogether) may be preprocessed in some other way and can be substituted for the background image. The second required image is an arbitrary visual mask whose shape forms the foveal window. For example, an aerial image of a runway is shown in Fig. 13, along with a Gaussian mask image ($\sigma = 100$). Both images are used in the following exemplar development of a GCD.

Using special-effects compositing terminology, the mask image simply constitutes the matte image which serves as the alpha mask for blending of the foreground (high-resolution) and background (low-resolution) images. The matte image is typically a normalized grey-scale image, where pixel values of 1 represent portions of the high-resolution image that show through while values of 0 are masked and therefore replaced by the corresponding background image pixels.

<table>
<thead>
<tr>
<th>Degradation Level</th>
<th>Image 1</th>
<th>Image 2</th>
<th>Image 3</th>
<th>Image 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2^1$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$2^2$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$2^3$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$2^4$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 13. “Original runway image with Gaussian mask with $\sigma = 100$.” [9]
To obtain a composited rendering of a foveal high-resolution window atop a low-resolution background, three textures are created for a quadrilateral. The first texture, assigned to texture-unit 0, or TU0, is the image mask. The second texture is the given image, which is assigned as the foreground image at texture-unit 1, or TU1. The third texture is the original image used for the foreground, also mipmapped, with the exception of the use of different LOD. It is the coarser LOD that generates the degraded background in the GCD. During display, the mask texture at TU0 is translated to the real-time coordinates of the foveal position. The process is shown diagrammatically in Fig. 14, with the callout showing the change in resolution between foveal and background regions.

I. Fragment Programming

The three-texture approach described previously leads to a biresolution display. For a more accurate representation of human visual acuity, multiple levels-of-detail are needed in the periphery, resulting in anisotropic peripheral degradation. To provide multiple levels of resolution in the periphery, the aforementioned multitexturing approach would require the use of multiple texture units. What is required is schematically shown in Fig. 15. At any given pixel concentrically related to the foveal position, a lookup is needed to a pixel at a specific level of resolution. Fragment programs provide just this type of flexibility by providing control of mipmap LOD bias at each fragment (pixel). The resulting sample is mapped to RGBA and written to the resulting vector. Unlike multitexturing, this rather elegant approach does not require explicit blending. Instead, the appropriate mipmap-level bias is obtained directly at each fragment. Note that if the degradation map is allowed to change dynamically, fragment programming allows dynamic visual field representation, such as, allowing multiple “regions of interest” (ROIs) which could be used for preattentive display purposes [14].

GLSL for computing spatiochromatic degradation at gaze point from an arbitrary 4-channel degradation texture map (courtesy of Duchowski and Çöltekin, 2007):

```
uniform float min_lod;
uniform sampler2D img_tex, deg_tex;

void main (void)
{
  // rgb→lum coefficient vector (from Foley et al. [16])
  vec4 lum = vec4 (0.299, 0.587, 0.114, 1.0);

  // fetch degradation texture sample
  vec4 deg = texture2D (deg_tex, glTexCoord[1].st);

  // invert lod mapping
  float lod = (1.0 − deg.w) * min_lod;

  // fetch lod biased image texture sample
  vec4 rgb = texture2D (img_tex, glTexCoord[0].st, lod);

  // return final composite
  gl_FragColor = vec4 (rgb.xyz * deg.xyz, rgb.w) + dot (lum.xyz, (rgb.xyz * (1.0 − deg.xyz)));
}
```

The GPU-based GCD code has been tested via both mouse- and eye-controlled foveal windows and runs well above hardware display rates (60 frames per second (fps)); note that display updates as late as 60 ms after eye-movement completion do not significantly increase the detectability of image blur and/or motion transients due to the update [24]). The code has also been extended to display video streams by interfacing with a video loading library. Due to hardware-assisted subsampling of a given image, Duchowski and Çöltekin [9] have found that the GPU-based GCD is sufficiently fast for real-time video degradation (display rates have also informally been measured well above 60 fps). This suggests that for gaze-
contingent display, image processing no longer poses a significant bottleneck, obviating the need for image preprocessing or storage.

IV. EYE TRACKING TECHNOLOGY

The following sections review the Ref. [10].

The above multitexturing and fragment programming techniques for gaze-contingent viewing are presented independent of eye tracker software. To fully implement a GCD, all that is necessary is to equip the main rendering loop with code that obtains the instantaneous x, y coordinates of the user’s gaze and applies these to the required translation of the foveal mask.

Eye tracker technology has advanced significantly since its modern inception in the early 20th century. From the first method of eye tracking using corneal reflection in 1901, through the use of contact lenses in the 1950s, today’s eye trackers generally employ analog video-based eye tracking techniques developed circa the 1970s [13].

Consider eye trackers within the following taxonomy:
1) First generation: eye-in-head measurement of the eye consisting of techniques such as scleral contact lens/search coil, electro-oculography;
2) Second generation: photo- and video-oculography;
3) Third generation: analog video-based combined pupil/corneal reflection.

The most salient form of eye tracking output is estimation of the projected Point Of Regard (POR) of the viewer. First and second generation eye trackers generally did not provide this type of information. So-called video-based combined pupil/corneal reflection eye trackers easily provide POR calculation following calibration, and are today de rigeur. Due to the availability of fast analog video processors, these third-generation eye trackers are capable of delivering the calculated POR in real-time. However, eye tracking technology is about to undergo its next evolution. Fourth-generation eye trackers, which began to appear on the market, are starting to make use of digital optics. Coupled with on-chip Digital Signal Processors (DSPs), eye tracking technology stands to significantly increase in usability, accuracy, and speed while decreasing in cost.

The state of today’s technology can best be summarized by a brief functional comparison of equipment available about 5 years ago and today’s state-of-the-art given in Table II.

<table>
<thead>
<tr>
<th>TABLE II</th>
<th>“FUNCTIONAL EYE TRACKER COMPARISON”[10]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technology</td>
<td>legacy systems</td>
</tr>
<tr>
<td>Calibration</td>
<td>analog video</td>
</tr>
<tr>
<td>Optics</td>
<td>5- or 9-point, tracker-controlled</td>
</tr>
<tr>
<td>Communication</td>
<td>requires focusing thresholding</td>
</tr>
<tr>
<td>Synchronization</td>
<td>serial</td>
</tr>
<tr>
<td>API callback</td>
<td></td>
</tr>
</tbody>
</table>

In general, most eye tracking applications perform the following [13]:
1) Connection: establish connection with the eye tracker (e.g., serial port or TCP/IP).
2) Calibration: display calibration points at the appropriate location and time.
3) Synchronization: display stimulus at the appropriate time (the eye tracker should be able to inform the application program of its state, or vice versa).
4) Data streaming: use eye tracker to capture data and/or update the stimulus scene in a gaze-contingent manner.

An example of a fourth-generation eye tracker is available from [34]. The Tobii 1750 eye tracker can be configured in several ways, one of which is acting as a server for a (possibly remote) eye tracking client application. The benefit of this organization is platform independence since communication between client and server occurs over TCP/IP. Platform independence is true for an eye tracker communicating via a serial cable as well, although serial communication requires relatively closer proximity between the eye tracker and application computer. An example configuration with an application computer (e.g., Linux PC) connected to the Tobii eye tracker is shown in Fig. 16.

V. CONCLUSION AND DISCUSSION

This article reviewed and presented current hardware-accelerated techniques for real-time simulation of arbitrary visual fields over still images and video suitable for a GCD. The main goal of this contribution is to alert the eye-tracking, vision sciences, and perceptual graphics communities with available computer graphics techniques facilitating the investigation of fundamental processes of visual perception. The banality of the solution is a consequence of hardware catching
The given hardware-accelerated fragment programming technique offers considerable flexibility for future perceptual and graphics research. A rather powerful but as yet unexploited benefit of fragment programming in this context is the potential for gaze-contingent color degradation. This is achieved by the use of a 4-channel degradation mask. Since only one channel (the alpha channel) is needed for resolution degradation, it is natural to use the remaining RGB channels to represent color degradation maps. Color degradation can be independently controlled in RGB color-space, since each of the RGB channels is itself a normalized image. Image color can simply be degraded by scaling a given pixel’s color by a scalar found in the corresponding degradation image RGB channels. A pixel’s output color is then interpolated between the pixel’s full color (original) and its luminance, where luminance is obtained from a constant conversion coefficient vector [16]. Due to the independence of the RGB degradation channels, this offers a rather powerful technique for exploring perceptual effects of peripheral color degradation (see Fig. 17). While peripheral visual acuity (and contrast sensitivity, e.g., see Reddy [31] and Luebke et al. [25]) has been studied widely, peripheral color sensitivity (and degradation) has not. Newly developed color degradation metrics, for example, following the classic evaluation paradigm of Funkhouser and S’equin [18], could affect perceptual rendering of images, games, and the like. It is plausible that a perceptually-based color degradation metric can be empirically derived to accelerate global illumination algorithms, in manners analogous to the adaptation or contrast-sensitivity examples of Ferwerda et al. [15] and Ramasubramanian et al. [30], respectively. Faster global illumination algorithms, for example, ray tracing or radiosity, may in turn lead to more efficient production of computer-generated imagery.

Fig. 17. “Real-time spatiochromatic degradation of Lena.”[9]

REFERENCES


