Gaze-dependent Ray Tracing

Adam Siekawa*
Supervised by: Radoslaw Mantiuk†

Institute of Computer Graphics
West Pomeranian University of Technology
Szczecin / Poland

Abstract

In this paper we introduce a method for acceleration of the real time ray tracing by using characteristic traits of visual perception. Ray tracing is a demanding rendering technique which is much slower than currently dominating scanline methods. Performance hit especially arise when we use huge amount of samples for anti-aliasing or other sample-based effects. We show how to decreases number of rays by increasing the perceptual size of selected pixels by using combination of eye tracking and the human gaze-dependent contrast sensitivity. Our study shows that number of processed pixels can be reduced three times without perceptually noticeable quality loss. As a result, we can greatly increase performance of ray tracing.

1 Introduction

The gaze-dependent vision is a characteristic way in which the human visual system builds an image of the world. We perform frequent and rapid eye movements, called saccadic movement, and follow moving objects in the smooth pursuit movement [2]. These rapidly changing snapshots are combined by the Human Visual System (HVS) in a stable image of the entire scene. Interestingly, a gaze-dependent model of image synthesis is not used in contemporary computer imaging systems, even despite the fact that a significant reduction of computation complexity is possible during image rendering in the parafoveal and peripheral regions of vision.

Ray tracing is a popular image synthesis technique which can benefit from the gaze-dependent characteristic of vision. Generally, even using the basic Whitted model [11], the ray tracing can produce images of the higher quality that the scanline techniques. However, this is achieved at the expense of larger computation complexity. The main bottleneck of this technique - finding intersections - can be reduced by decreasing the number of primary rays. In this work we propose a gaze-dependent ray tracing in which the number of rays per unit angle fits the sensitivity of HVS. We use the gaze-dependent contrast sensitivity function (CSF) to reduce sampling in peripheral vision. Temporal location of the gaze point is captured by the eye tracker and used by the interactive ray tracing system to render images with the highest quality only in the gaze-point surrounding.

In Sect. 2 we outline the directionality of the human vision, introduce the gaze-dependent CSF and discuss existing gaze-dependent techniques of image synthesis. Sect. 3 presents our gaze-dependent ray tracing system based on the weighted sampling. In Sect. 4 we show how the gaze-dependent sampling was implemented in our ray tracer. Sect. 5 discusses the achieved performance boost with respect to image quality deterioration. The paper ends with conclusions and future work in Sect. 6.

2 Background

In this Section we present a basis of the human eye physiology and describe technologies used in the gaze-dependent rendering frameworks.

2.1 Gaze-dependent contrast sensitivity function

Human vision has a strong feature of the directionality of view. We can see the details only in a small viewing angle subtended 2-3 degrees of the field of view. In this range, a human sees with a resolution of up to 60 cycles per angular degree, but for a 20-degree viewing angle, this sensitivity is reduced ten times [6].

The fundamental relationship describing the behaviour of the human visual system is the contrast sensitivity function (CSF) [1]. It shows the dependence between the threshold contrast visibility and the frequency of the stimulus. The CSF can be used to e.g. better compress the image by removing the high frequency details that would not be seen by humans. An extension of the CSF, called the gaze-dependent CSF, is measured for stimuli observed in various viewing angles [3, 12]. It models the impact of deviations from the axis of vision (called eccentricity (E)) to the most recognisable stimulus frequency (see Fig. 1).

In this work we use the gaze-dependent CSF proposed by Peli et al. [3]:

$$C_t(E,f) = C_t(0,f) * exp(kfE), \qquad (1)$$

*adamsiekawa@gmail.com
†rmantiuk@wi.zut.edu.pl

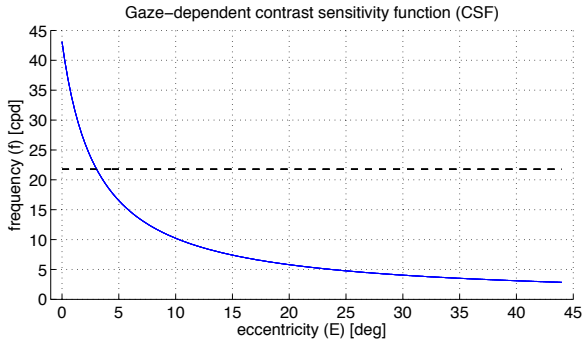

Figure 1: Gaze-dependent contrast sensitivity function. The dashed line shows the maximum frequency of our display.

where C_t denotes contrast sensitivity for spatial frequency f at an eccentricity E, k determines how fast sensitivity drops off with eccentricity (the k value is ranged from 0.030 to 0.057). Based on the above equation, the cut-off spatial frequency f_c can be modelled as:

$$f_c(E) = min(max\_display\_cpd, E_1 * E_2/(E_2+E)), \quad (2)$$

where E_2 is retinal eccentricity at which the spatial frequency cut-off drops to half its foveal maximum (it ranges from E_1=43.1 to 21.55), and $E_2 = 3.118$ (see details in [13]). An example region-of-interest mask computed for our display based on the gaze-dependent CSF is presented in Fig. 2. Applying this mask, one can e.g. sample an image with varying frequency generating less sampling rays for the peripheral regions of vision.

Figure 2: Region-of-interest mask computed based on CSF for an image of 1920x1080 pixel resolution (gaze position at $(1000, 500)$), lighter area denotes higher frequency of HVS.

2.2 Gaze-dependent image synthesis

Information about temporary gaze direction was previously used to reduce the computational complexity of the image synthesis. An example of this approach is the technique called the level of detail (LOD), in which the simplified models of objects are located in the peripheral areas of vision [7, 9].

In the ray casting [9] and volumetric rendering [5] the gaze-dependent sampling is applied in the screen space.

A similar solution was used to accelerate the ambient occlusion algorithm [8]. This technique introduces a novel filtration method, in which the global lighting is calculated only for the area surrounding the gaze point. In peripheral areas of vision only fast computations based on the local lighting model are performed. The perceptual experiments showed that the participants did not notice the quality deterioration of the generated images.

The models of the gaze-dependent vision seems to gain an increasing importance in improving the efficiency of the image synthesis. The leading IT companies are interested in new gaze-dependent rendering techniques. For example, in the solution proposed by Gunter and others in [4], the scanline-based rendering engine generates three low-resolution images corresponding to the different fields of view. Then, the wide-angle images are magnified and combined with non-scaled image of the area surrounding the gaze point. Thus, the number of processed pixels can be reduced by 10-15 times.

3 Gaze-dependent rendering

Fig. 3 presents the gaze-dependent rendering scheme. Our system requires the eye tracker data which represents a momentary gaze direction of a human observer. We render the scene using ray tracing. The screen space is sampled (the primary rays are generated) according to the gaze-dependent contrast sensitivity function. Less rays is generated in parafoveal and peripheral regions. The output image is reconstructed from the non-uniformly distributed samples and displayed in real time on the screen.

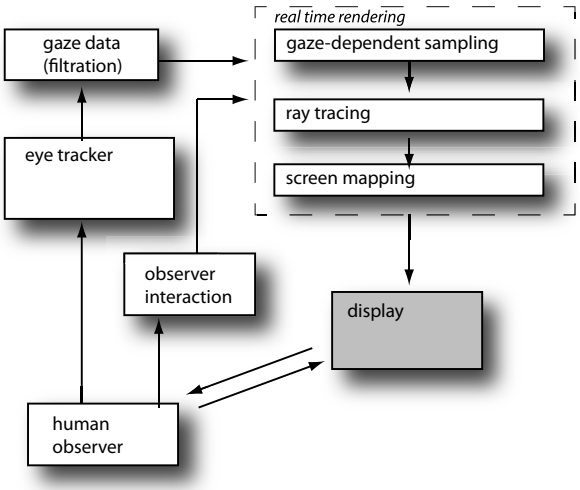

Figure 3: Gaze-dependent rendering system.

The whole system must be scaled in the real-world di-

mensions. We transform the gaze data to screen space using physical dimensions of a display, its resolution, and viewing distance between observer and the display screen.

3.1 Gaze-dependent sampling

The gaze-dependent CSF defines a solid angle in which a human cannot see details. This angle defines a limit of the HVS resolution and can be scaled in the perceptual JND units. We call this angle a perceptual unit angle. The further from gaze point a sample is, the larger the angle becomes. In this angle the human eyes integrate the image, i.e. it computes the average luminance. To sample an image during rendering, we use the constant number of rays per perceptual unit angle. For peripheral vision, the perceptual unit angle covers more pixels and the number of rays per image area decreases (see Fig. 4).

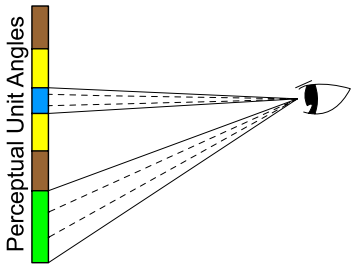

Figure 4: Less sampling rays falls on the area in the peripheral vision. The perceptual unit angles are marked in colours.

The perceptual unit angle (α) covers an area derived from CSF:

$$\alpha = 1/(2 * E_1 * \frac{E_2}{E + E_2})[deg], \quad (3)$$

where E denotes a viewing angle subtended from the gaze direction to the direction towards a considered pixel, E_1=43.1, E_2=3.118 [3]. This angle can be computed using equation:

$$E = atan(\frac{p_{distance} * p_{size}}{d})[deg], \quad (4)$$

where $p_{distance}$ is a distance between pixel and the gaze point in [pixels], p_{size} is a physical pixel size in [cm], and d denotes a distance from the screen to observer's eyes expressed in [cm].

The number of pixels covered by a perceptual unit angle α can be derived from:

$$\rho = \|\frac{\alpha}{\beta}\|, \quad (5)$$

where β is viewing angle in [deg] corresponding to one pixel.

In the gaze dependent renderer one can reduce the number of rays shoot per pixel based on the information whether a considered pixels belongs to the larger or smaller perceptual unit angle.

In our system we group together pixels belonging to one perceptual unit angle and form cells. Then, the image is sampled based on distribution of the cells. We shoot the constant number of anti-aliasing rays per cell (see details in Sect. 4.2) but, as the cells are larger in peripheral vision, the total number of sampling rays is reduced. Cells positioned further from gaze point will produce less anti-aliased results, however the artefacts will not be visible for the human observer. Cells are put together into an image after rendering. Visual representation of cell distribution is presented in Fig. 5.

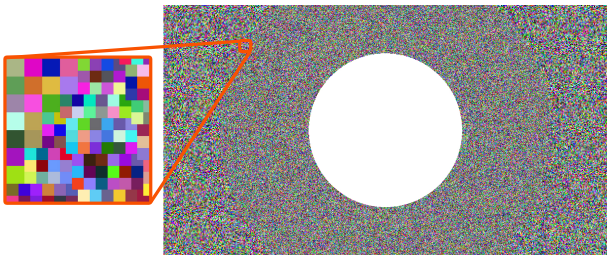

Figure 5: Cell map generated for an example location of the gaze point. Each cell larger than original pixel size is given a random color, non scaled cells are coloured in white. The enlargement shows unique structure of the map.

3.2 Rendering

The ray tracing was used for rendering because of simplicity of implementation of complex sampling schemas in the screen space. We implemented the Whitted ray tracing model which supports Phong lighting, shadows, reflection and refraction rays. See the implementation details in Sect. 4.1.

In this work we use a prototype renderer which does not work in real time. This solution does not meet the main assumption of the gaze-dependent rendering system, i.e. the gaze-driven rendering in which image content is changed with the gaze movement. However, our setup allows to perform the quality tests based on the offline results.

In future work we plan to adapt a real time ray tracer, such as OptiX [10] or Octane Renderer. Alternatively, we consider implementation of own ray tracer engine based on OpenCL or CUDA APIs. In the raw estimation, one ray should be rendered in $3e^{-8}$ [sec] to generate 60 frames-per-second in a typical viewing conditions. This requirement seems to be a challenge for a typical ray tracer and the gaze-dependent solution which significantly reduces the number of traced rays is highly beneficial (see details in Sect. 5).

4 Implementation

We implemented our own ray tracer extended with the gaze-dependent sampling technique. All images presented in this paper were rendered with this application.

4.1 Ray tracer

Ray tracer that was used in our project as a proof-of-concept is an off-line renderer implemented in C Sharp. It is build in a content based fashion, where one can create material by adding extra effects to the base type. There are two lighting models implemented: Phong and Ashikhmin-Shirley models. The ray tracer supports reflections, refractions, textures, and both hard and soft shadows.

One can load 3D scene stored in most of the popular formats, e.g. Wavefront OBJ format, COLLADA, or Autodesk 3DS file. The ray tracer can also render non-triangulated spheres. A scene is created in a code, i.e. one can position loaded models, created spheres, lights and camera then append them to the rendering list. The octree acceleration structure is applied to improve performance. Moreover, C# style *parallel foreach* is used for sampling each cell individually and utilise all available CPU cores. Results are saved as a linearly tone-mapped bitmap image. It is also possible to output image sequence which can later be put into a video.

Our ray tracer implements stochastic, regular (samples are distributed in a grid fashion) and adaptive anti-aliasing techniques. However, we found the most useful the stochastic sampling based on the random samples distribution. We use this type of anti-aliasing in all tests. Sample rays are distributed to fit the whole cell region (see details in 4.2). The first ray is always shoot in the centre of the cell (pixel or group of pixels). For the following samples we generate random single precision value which is used for offsetting ray direction.

4.2 Cell map generator

A cell map generator is an implementation of the gaze-dependent sampling in which cells are the discrete representation of the perceptual unit angles (see Sect. 3.1). One cell can cover one or more pixels, as seen in Fig. 6. Our algorithm requires information about gaze point (obtained from eye tracking device) and viewing conditions (width, height and viewing distance from a display) to compute number of pixels ρ that is within perceptual unit angle size (see Sect. 3.1). Result of the cell map generator is a *cell vector* with one cell per one unit angle and a *cell mask* which stores relationship between cells and pixels.

Single cell is a structure described by the set of parameters:

- unique cell id stored also in the cell mask
- size ρ, when equal to 1 it indicates that cell is covering single physical pixel.

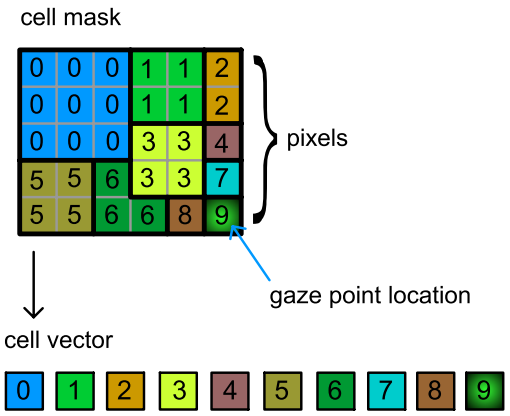

Figure 6: A cell vector for a 5x6-pixel image. Groups of pixels covered by the unit angle computed for a current distance from the gaze point (pixel with index of 9) are assigned to consecutive indices in the cell vector.

- pixel's centre position in the screen space
- camera information
- default luminance value (clear color)

The *cell id* is necessary for image reconstruction. *Size*, *position* and *camera data* is used during ray tracing procedure to generate the primary rays. The cell mask forms a matrix (with the size of destination image), which contains *cell ids* and helps to maintain the overlapping cells.

The cell vector is sent to ray tracing pipeline and is used during AA rays generation. We distribute the constant number of samples in a region covered by a given cell.

Final step is image reconstruction, for that we need to use cell mask mentioned earlier. As illustrated in Fig. 6, pixels have the same *cell id* as the cell that covers them. In order to retrieve our image, we need to iterate over that mask and extract the final color value from a cell with the same *cell id* and write it into a place holder for an image (e.g. DirectX or OpenGL texture).

5 Results

We rendered a set of images applying the gaze-dependent sampling calibrated for our hardware setup: 1080p resolution display measuring a 50 [cm] screen width and 35 [cm] height, observed from 60 [cm]. We used 32 samples for the stochastic anti-aliasing. This number seems to generate almost perfectly anti-aliased images of our test scene. The computational complexity remains the same as in classic ray tracing and cell map generation is not taken into account since it is used as a precomputed input. Example renderings are presented in Fig. 7. In the top image a typical ray tracing technique with the per-pixel stochastic anti-aliasing was used. The bottom image was generated

using the gaze-dependent technique with 32 anti-aliasing rays per cell (perceptual unit angle). The quality of image with reduced number of samples is noticeably worse, however this deterioration is not visible if observer is looking at the gaze point. This phenomena is even better visible on video we delivered in the supplementary materials. We prepared a HDTV clip (1080p, 25 FPS) with sampling rate reduced to 4 anti-aliasing rays per a cell.

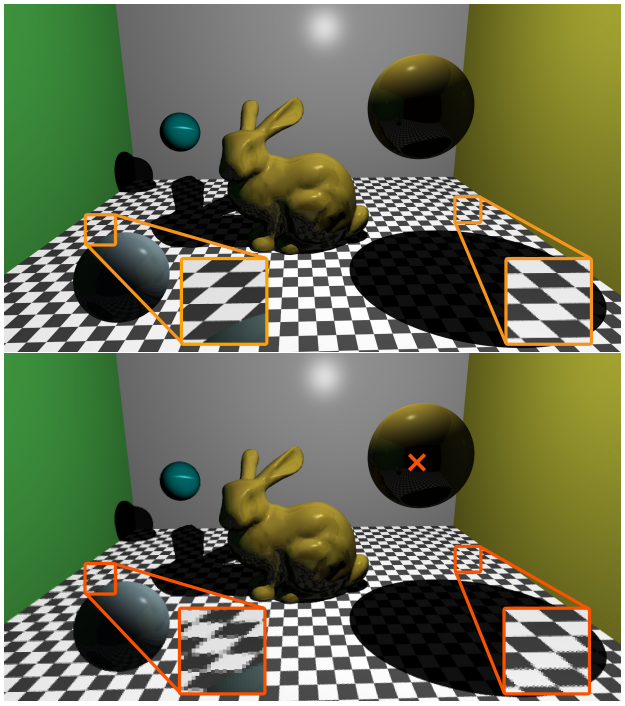

Figure 7: Comparison of the full frame (top) and the gaze-dependent rendering (bottom). Gaze point is marked as the red cross in the bottom image. The enlargements on the right depict borders between region with $\rho = 1$ and $\rho = 2$. Left enlargements show artefacts in region far away from the gaze position.

Cell overlapping

The cell mapping produces more apparent artefacts with increasing distance from the gaze point. Some of the cells may overlap each other, creating characteristic shapes similar to letter 'L', which are visible in Fig. 5. These artefacts are appearing when cell of size $\rho = n$ neighbours cell with size of $\rho = n - 1$, causing displacement of each consecutive cell. However, observer couldn't see this artefacts.

Performance

We measured the rendering performance on the laptop with Intel i3-2310M CPU, 2.1 GHz with 2 cores, 4 threads in total. It took 48 minutes and 16 seconds to render full frame anti-aliased image (see Fig. 7,top). During this time a 66.35 million anti-aliasing rays were traced. The cell map method needed only 14 minutes and 52 seconds with 20.65 million anti-aliasing rays shoot. The gaze-dependent technique was more than 3 times faster and almost 70% of sampling rays was required.

The acceleration will be even more significant for future displays of the retinal resolutions (60 cycles per visual angle). Our display should have a resolution of 5400x3900 pixels to reach the HVS resolution. In this case a typical full frame ray tracing would require 674 million sampling rays, but with cell map approach we would need only 27 million million rays, which is around 95% less.

6 Conclusions and Future Works

In this work we introduced gaze-dependent rendering as a sample reduction method for increasing ray tracing performance. Our algorithm is based on gaze-dependent CSF. It takes into account viewing conditions and physical dimensions of the display. We demonstrated how the cell mapping algorithm based on perceptual gaze-dependent sampling of the screen space can result in major performance boost. Although presented algorithm generates artefacts in the parafoveal region, they are unnoticeable for a viewer. In the paper we mainly focus on improving performance by accelerating anti-aliasing algorithms, but we expect that the same concept can be applied to other performance heavy algorithms based on sampling.

In future work we plan to deploy a real-time version of the system. In addition to the implementation of a fast ray tracker, our cell map generation process might proof to be difficult for parallel computing. One way of solving this issue is creating a precomputed cell map, which would use extra memory (four time more than map generated during runtime). We also want to address the problem of overlapping cells. Our algorithm might also proof useful in increasing performance of other rendering techniques i.e., path tracing or photon mapping.

References

[1] P. G. J. Barten. *Contrast sensitivity of the human eye and its effects on image quality*. SPIE Press, 1999.

[2] Andrew T. Duchowski. *Eye Tracking Methodology: Theory and Practice (2nd edition)*. Springer, London, 2007.

[3] Jian Yang Eli Peli and Robert B. Goldstein. *Image invariance with changes in size: the role of peripheral contrast thresholds*. JOSA A, Vol.8, Issue 11, 1991.

[4] Brian Guenter, Mark Finch, Steven Drucker, Desney Tan, and John Snyder. Foveated 3d graphics. *ACM Trans. Graph.*, 31(6):164:1–164:10, 2012.

[5] Marc Levoy and Ross Whitaker. Gaze-directed volume rendering. In *Proceedings of the 1990 symposium on Interactive 3D graphics*, I3D '90, pages 217–223, New York, NY, USA, 1990. ACM.

[6] L. C. Loschky, G. W. McConkie, J. Yang, and M. E. Miller. The limits of visual resolution in natural scene viewing. *Visual Cognition*, 12:1057–1092, 2005.

[7] David P. Luebke and Benjamin Hallen. Perceptually-driven simplification for interactive rendering. In *Proceedings of the 12th Eurographics Workshop on Rendering Techniques*, pages 223–234, London, UK, UK, 2001. Springer-Verlag.

[8] R. Mantiuk and S. Janus. Gaze-dependent ambient occlusion. *Lecture Notes in Computer Science (Proc. of ISVC'12 Conference)*, 7431(I):523–532, 2012.

[9] Hunter A. Murphy, Andrew T. Duchowski, and Richard A. Tyrrell. Hybrid image/model-based gaze-contingent rendering. *ACM Trans. Appl. Percept.*, 5:22:1–22:21, February 2009.

[10] Steven G. Parker, James Bigler, Andreas Dietrich, Heiko Friedrich, Jared Hoberock, David Luebke, David McAllister, Morgan McGuire, Keith Morley, Austin Robison, and Martin Stich. Optix: A general purpose ray tracing engine. *ACM Transactions on Graphics*, August 2010.

[11] Turner Whitted. An improved illumination model for shaded display. *Graphics and Image Processing*, 23(6):343–349, 1980.

[12] J. Yang, T. Coia, and M. Miller. Subjective evaluation of retinal-dependent image degradations. In *Proceedings of PICS 2001: Image Processing, Image Quality, Image Capture Systems*, Society for Imaging Science and Technology, pages 142–147, 2001.

[13] J. Yang, X. Qi, and W. Makous. *Zero frequency masking and a model of contrast sensitivity.* Vision Research, 1995.