A Gaze-Contingent System for Foveated Multiresolution Visualization of Vector and Volumetric Data

Thanawut Ananpiriyakul (SnapLogic), Joshua Anghel (Amazon), Kristi Potter (NREL), Alark Joshi (USF); Department of Computer Science, University of San Francisco, San Francisco, CA, USA

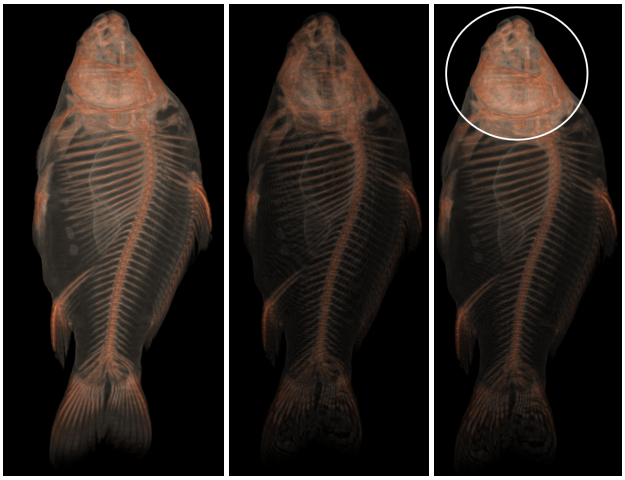
Abstract

Computational complexity is a limiting factor for visualizing large-scale scientific data. Most approaches to render large datasets are focused on novel algorithms that leverage cutting-edge graphics hardware to provide users with an interactive experience. In this paper, we alternatively demonstrate foveated imaging which allows interactive exploration using low-cost hardware by tracking the gaze of a participant to drive the rendering quality of an image. Foveated imaging exploits the fact that the spatial resolution of the human visual system decreases dramatically away from the central point of gaze, allowing computational resources to be reserved for areas of importance. We demonstrate this approach using face tracking to identify the gaze point of the participant for both vector and volumetric datasets and evaluate our results by comparing against traditional techniques. In our evaluation, we found a significant increase in computational performance using our foveated imaging approach while maintaining high image quality in regions of visual attention.

Introduction

The interactive visualization of large-scale, high-resolution data is often hindered by the computational costs of rendering algorithms coupled with considerable data sizes. While algorithmic optimizations and innovative data management strategies can greatly improve run time rates, data size and complexity will continue to challenge our ability to present data in real time. In this work, we propose a novel method to increase the efficiency of rendering large datasets by strategically reducing the resolution of the data display based on the inherent characteristics of the human visual system. Rather than render a full resolution data set across the entirety of the display, we leverage advances in tracking technology to adaptively mimic the multiresolution behavior of the human eye to accelerate the interactive rates of large-scale vector and volume visualization. We have developed a system that produces high-quality, interactive visualizations of large-scale data that limits the resolution of the data display outside the viewer's gaze and thus achieves high frame rates while maintaining computational efficiency. As shown in Figure 1, our foveated rendering techniques combine the strength of high- and medium-quality rendering to provide exceptional resolution only in areas of visual attention.

The human visual system is one of the most complex systems in the human body, with a network of highly specialized cells that process what we see in stages [LH08]. The *fovea* is the central region of this system and is responsible for sharp, central vision. The high spatial acuity of the fovea is due to the dense packing of cone cells in this area, and acuity drastically falls off towards the periphery as the number of cone cells also drops, as shown in Figure 2. *Foveated imaging* [GP98] leverages the nuances of spatial acuity to dedicate computational resources to areas where the visual system is most sensitive. Exploiting the fact that the spatial resolution of the human visual system decreases dramatically away from the point of gaze, foveated imaging allows for the real-time processing of images, videos, and data in a way that is seamless to the end user, but provides a significant improvement in the performance of underlying applications.



(a) High-Quality (b) Medium-Quality (c) Foveated

Figure 1. *Gaze-contingent rendering enhances computational speeds by rendering only important regions of a data set, determined by the viewer's focus, at high-resolutions, and all other areas at lower-quality. From left to right, high quality rendering provides detail, but is computationally demanding, medium-quality rendering increases computational speeds, at the cost of visual resolution, while gaze-contingent rendering displays only the region at the center point of gaze of the viewer in high-detail.*

In this paper, we present a *gaze-contingent* multiresolution system that blurs and scales rendered images to facilitate foveated visualization with smooth transitions from highly-focused to low-resolution regions. Our primary contributions include the use of face-tracking to drive adaptive-resolution visualizations, and the modifications to well-known vector and volume rendering algorithms to create compelling and computationally efficient imagery. We demonstrate our foveated visualization system on large-scale spatio-temporal vector and volumetric data and conduct a performance analyses that shows a 2-2.5X frame rate improvement on interactive explorations.

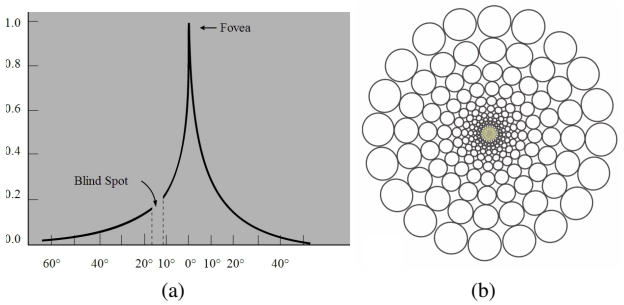


Figure 2. *Left, a graph of visual acuity showing how it decreases rapidly as the distance from the fovea increases. Right, an illustration of the resolution of our visual field. It is higher in the center but much lower in the edges (peripheral vision). Image credits: Livingstone and Adams [LH08] (left) and Johnson [Joh13] (right).*

Related Work

Our system builds upon previous work on gaze-contingent displays to focus the computational efforts to a specific region of interest. Using importance-based volume and vector visualization techniques, rendering is changed across the display in concordance with our user-driven region of concern. This type of display is similar to focus plus context methods which also change rendering across a display to highlight features within a data set. In contrast to our work, these features are often automatically discovered through algorithms such as clustering or directly chosen by the user, rather than based directly on the viewer's gaze.

Foveated Imaging

The focus of much foveated imaging work is on reducing the spatial resolution of transmitted video images with minimal effect on perceived image quality. Geisler and Perry [GP98] introduced a multiresolution system for low-bandwidth video communication and demonstrated that foveated videos can preserve visual quality much better than compressed videos. Their follow-on work produced higher quality videos and worked for variable resolution displays [PG02]. Chang and Yap [CY97] demonstrated a wavelet-based approach to generating foveated images through an adaptive layout that weighed the blending functions for seamless images. We use blending functions as well to ensure a smooth transition for the viewer. The EyeRIS system [SRIR07] was one of the first eye-movement contingent display (EMCD) systems and facilitated near-real-time interaction with images through a combination of custom hardware and software. A comprehensive review on gaze-contingent multiresolution displays can be found in Reingold et al. [RLMS03].

In the 3D graphics and virtual reality domains, foveated rendering is becoming increasingly popular. Guenter [GFD*12] applied foveated imaging to the rendering of traditional 3D scenes and found a 5–6x improvement in the overall graphics performance. Patney et al. [PSK*16] used a perceptually-improved foveated system to generate a perceptual target to reduce artifacts for interactive foveated rendering. They introduce a novel algorithm that performs temporal anti-aliasing to address problems associated with saccades and contrast enhancement to provide details that can be perceived by the human peripheral system but may be lost due to filtering.

Despite the important benefits offered by eye-movement contingent display control to many areas of vision research, a major limiting factor has been the unavailability of general-purpose systems that enable flexible gaze-contingent manipulation of the stimulus [SRIR07]. A second limiting factor has been the difficulty of guaranteeing real-time performance to ensure an upper bound on the delay between the occurrence of oculomotor events and an update of the stimulus on the display. For these reasons, we turned to *face-tracking* as a more robust method for identifying user gaze.

Focus-Driven Volume and Vector Visualization

Methods for displaying 3D volumetric and vector flow data sets are common in the visualization community and include volume rendering [DCH88], streamlines [WS01, ZSH96, LKJ*05, HWHJ99, CYY*11], glyphs [WSF*95], and line-integral convolution (LIC) [CL93, SJM96]. These techniques are often modified to improve performance by selecting regions of importance to display with more fidelity (importance-rendering) or to provide context to selected regions of interest (focus plus context). The work most closely related to this work is foveated volumes by Yu et al. [YCHZ05]. In contrast to our work, the boundaries for the foveated region in the volume are clearly distinct, whereas our focus is to provide a seamless transition between the foveated region and the surrounding peripheral imagery.

For vector visualization, Schroeder et al. present an interface for creating streamline visualizations of varying level-of-detail based on the user's input [SCK10] using a gesture-based interface. Focus and context has been added to vector fields through a deformation-based approach based on partitioning the volume space of the flow field into blocks that are then deformed to reposition existing streamlines [TWSK14]. Importance fields have also been incorporated into time-varying vector fields through tools that allow users to explore large datasets and highlight features of interest seamlessly [WYM08]. Telea et al. [TVW99] presented a tree-based method for varying the density of glyph placement to simplify regions of a vector field; however, the approach lacks the ability to provide smooth transitions between levels of detail in real-time. In contrast to our gaze-directed technique, the method uses analysis of the vector field data to determine which regions will be more or less simplified.

Automated importance-driven rendering to highlight features in a volumetric dataset was introduced by Viola et al. [VKG05]. Illustrative techniques [RE01, BG07] have also been explored to preserve context [BGKG06] and draw attention to features. Focus and context has been used in application domains for the visualization of complex vascular structures such as aneurysms [GNBP11] and in augmented reality applications [KMS07] to preserve context and to facilitate exploration of regions of interest. Interactive techniques to author and view cutaways of 3D models were introduced by Li et al. [LRA*07] and in the VolumeShop [BG05] system. These techniques let users author illustration-style imagery from volumetric data to draw attention to features in the data. There has also been significant research in applying lens-based mechanisms to explore data in interactive systems and to draw a viewer's attention in static images. Tominski et al. [TGK*17] provides an extensive survey of various systems that use the lens-based paradigm to visualize graphs, volumetric, spatio-temporal, flow, and multivariate data, as well as individual text and corpus data.

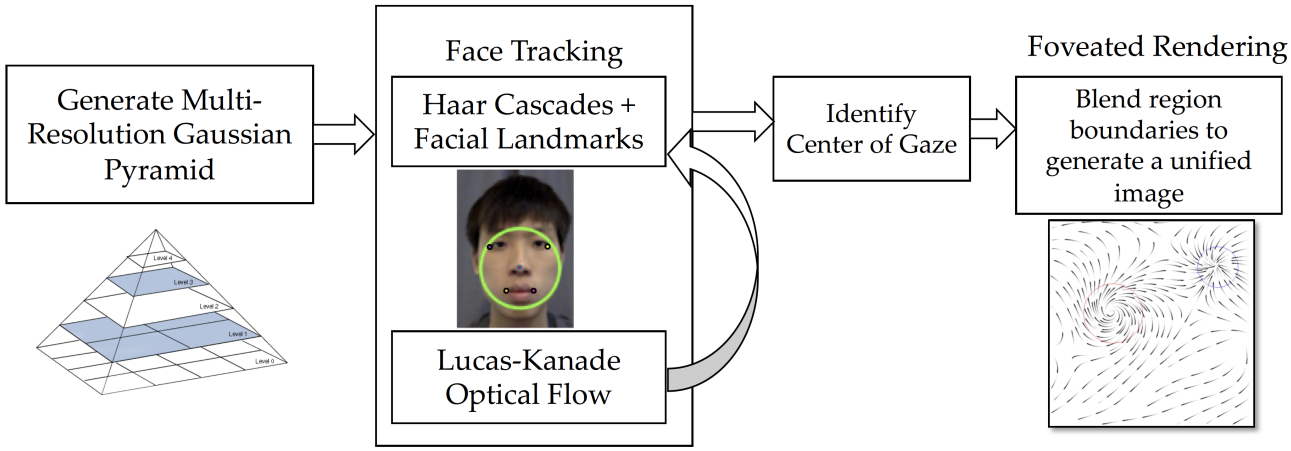


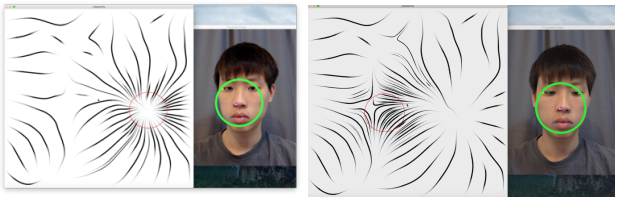
Figure 3. *This schematic conveys the overview of our system. We pre-compute a multiresolution Gaussian pyramid of the vector representations before we start gaze tracking. Gaze tracking is conducted using a combination of Haar cascades and facial landmarks in combination with the Lucas-Kanade Optical Flow algorithm. We use the face tracking to identify the center of the participant's gaze. We use the center to identify the region of interest. This is followed by the Foveated Rendering step where we blend the boundaries of the various multi-resolution representations used to generate a unified image.*

Approach

Our approach for improving the performance of volume and vector visualizations of large-scale datasets relies on incorporating *face-tracking* into the visualization algorithms to guide high-resolution rendering to only areas of importance. Figure 3 shows an overview of our system. The face-tracking system determines the gaze-point within the scene, and we then use that gaze-point to define the importance regions which, in turn, drives the multi-resolution rendering. Algorithmic changes to both vector and volume visualization techniques are required to incorporate the varying resolution. In addition, because of the varying resolution across a single image and the time-component of the data sets, temporal coherence becomes an issue. Failure to maintain coherence results in visual artifacts such as "popping" or flicker which can be distracting or even detrimental in understanding a data set.

Face Tracking

We turned to face tracking as it beneficial over eye-tracking since it does not succumb to the jittering effects of saccades. It is also a marker-less system and does not require the use of the eye-tracking hardware which can become cumbersome after prolonged periods of use. Our face-tracking system is implemented using commodity webcams. The Flandmark [UFH12] libraries are used for detecting facial landmarks and OpenCV [BK08] is used for reliable implementations of computer vision algorithms. To begin, we use a Haar Cascades frontal face classifier [BK00] to detect the face of the viewer and calibrate the system based on the centroid of the face and its position on the screen. Tracking the location of facial landmarks is done using Flandmark and specifically, we track the location of the eyes, nose, and mouth of the viewer. The Lucas-Kanade Optical Flow algorithm [BK00] is used to track facial landmarks as the viewer's face moves. If the user's faces moves too much, we re-run the classifier as shown in Figure 3. Similarly, to manage situations with varying lighting or when the facial landmarks may go off screen, we re-activate the Haar Cascades classifier and re-identify facial landmarks at regular intervals (approximately every 10 seconds or so). Figure 4 shows a viewer looking at different focus points on the screen. As can be seen in the figure, the centers of the circle move depending on where the viewer is looking. As the face moves, the center of interest moves to generate high resolution representations for that part of the screen.



(a) The gaze of a viewer looking at a critical point near the right side. (b) A change of gaze leads to detail being closer to the center.

Figure 4. *Face tracking. Here, a viewer is seen changing his direction of gaze which leads to the changes in position for the facial landmarks.*

Interactive Importance Fields

We use the results of the face-tracking system to define an *importance field* across the scene. An importance field is a scalar field that describes areas of significance within the data domain. In this case, the importance field is imposed over the space of the vector field or data volume. At each location, a value of importance is defined based on gaze and user input, and importance is represented as a normalized scalar value representing the desired information density at that location. For example, an importance value of 1 would dictate the highest density of visual display elements at that particular location, while a value of 0 will render the minimal number of elements.

To create the importance field, we use the tracked gaze direction to define *focus points*. A focus point is controlled by defining a position (p_f) describing the center of the point, a radius (r_f) describing the size of the important region, and a weight (w) describing the strength of contribution. A wave equation is used to combine these parameters into a measure of the total impor-

tance contributed by the focus point to the entire importance field. The wave equation is implemented as a 1D function that evaluates importance based on the distance (d_e) from the position of an evaluated point (p_e) to the center of the focus point (p_f), i.e. $d_e = p_f - p_e$. The importance value is highest at the center of the focus point and decays as the point moves further from the center.

The choice of wave equation greatly influences both the look and computational expense of the resulting visualization, and thus we provide the user with a collection of equations to choose from:

$$\text{Linear:} \quad \Phi(d_e) = \min(w - \frac{d_e}{r_f}, 0)$$

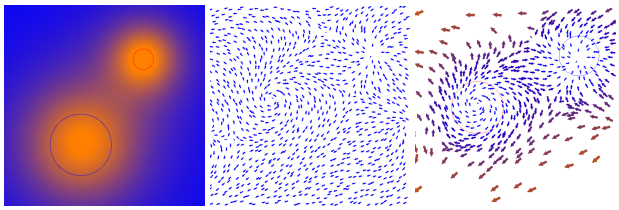
$$\text{Inverse:} \quad \Phi(d_e) = w\frac{1}{1+\frac{d_e}{r_f}}$$

$$\text{Inverse Square:} \quad \Phi(d_e) = w\frac{1}{(1+\frac{d_e}{r_f})^2}$$

$$\text{Gaussian:} \quad \Phi(d_e) = we^{-(\frac{d_e}{r_f})^2}$$

On a desktop platform, the choice of wave equation negligibly affected performance, and thus the Gaussian equation was chosen as the default, since it produces a smooth transition and visually appealing transition of importance.

The importance field is stored as a 2D texture, populated with importance values defined by sampling the wave equations of the defined focus points. Figure 5, left, shows an importance field resulting from two focus points, color mapped from lowest (blue) to highest (orange) importance. To optimize for performance, we only sample at the center of texture points and use bi-linear interpolation to acquire values at any continuous point. While a more accurate approach would directly evaluate the wave equations for any point in the field, our approximation favors reducing the number of evaluations to maintain interactivity.



(a) Importance field (b) Full resolution (c) Importance based

Figure 5. *An importance map, left, and its impact on a full-resolution glyph-based vector visualization, center, to create an importance-sampled vector visualization, right.*

User Interaction

While face-tracking is the main driver for creating our importance fields, we have discovered that allowing the user to manipulate focus points to also be useful. Examples of such interactions include adding focus points to allow for regional comparisons, removing saved points, or modifying the radius of focus points to accommodate large-scale or small mobile displays. In addition, focus points can be defined *automatically* by computing critical points within a vector data set, if desired. These computationally placed focus points can be used as a starting point for exploration of the data, and may be particularly useful for large, or highly-complex data sets that may take a substantial amount of user time to interrogate to find features of interest. Beneficially, adding in this capability to the system is trivial, and besides the algorithmic modifications described below, required only the implementation of simple keyboard or touch-based interfaces to achieve.

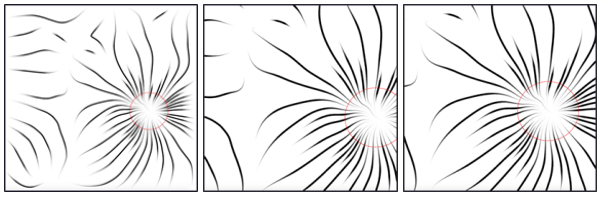


Figure 6. *Foveated Zooming to further focus in on the region of interest. The left figure shows the region of interest before zooming. The central figure shows the effect of naïve zooming on the center of the screen, whereas the right figure shows our foveated zooming where we zoom in on the region of interest as it is more relevant to the viewer.*

In addition to edits of focus points, we have also implemented *foveated zooming* which allows the user to zoom into the gaze-based region of interest rather than using the center of the screen as the zooming point. A user can use the keyboard to resize the region of interest on the screen and the mouse to rotate or zoom and pan the viewpoint as needed. Figure 6 shows a scenario where the left image shows the user identified region of interest that is followed by zooming in on it without foveated zooming (center figure) and with foveated zooming (right figure). Given that the viewer is fixated on that region, foveated zooming provides a viewer with more details in that region of interest.

Vector Field Visualization

Glyphs and streamlines are the most commonly used techniques for rendering vector fields. Both rely on the judicious placement of visual elements as clutter and temporal coherence can cause detrimental artifacts leading to poor efficacy of the technique and lowered understanding of the data. Importance fields are used to place more detail in regions of interest, be they a higher-density of glyphs or the seeding of more streamlines. However, care must be taken that the aforementioned visual artifacts are avoided.

Glyph Visualizations

Glyph-based representations of vector fields use strategically-placed symbols to indicate the direction and magnitude of a field. By varying the location and number of glyphs rendered in a particular region, the information density can be increased or decreased. To maintain interactivity as well as temporal coherence, a *glyph pool* is used to define the position and direction of the glyphs. A glyph pool is a pre-calculated high-density collection of potential glyphs, whose position and direction is fixed throughout the exploration. An example of a glyph field can be seen in Figure 5, center. During an interactive session, glyphs are sampled from the glyph pool, at rates reflecting the importance at each location. The choice of whether a glyph from the pool should be rendered is made using the glyph's *importance threshold*. The importance threshold is the minimum importance that must be present at the glyph's position for that glyph to be rendered.

Glyph Pool Sampling

Three methods for sampling the glyph pool and importance thresholds are investigated: random, grid-based and mip-mapped. *Randomly selecting* glyph positions and importance thresholds was explored with the intention of creating a uniform distribution of glyphs throughout the visualization. However, this approach results in glyph overlap and pockets of unintended density away from the focus points.

To alleviate the problem of occlusion, a *grid-based* sampling of glyphs was explored. Glyphs were placed on a uniform or jittered grid, which ensures glyphs do not occlude, as long as nearby glyphs are scaled appropriately and the jittering is small. We maintained the random assignment of importance thresholds. Even though glyph overlap is suppressed, the random thresholds still produce pockets of unintended densities.

To create a uniform spatial distribution of importance thresholds, we use a *mipmapping-inspired approach* to assign importance thresholds. This approach ensures that no two neighboring glyphs have low importance values and thus avoids pockets of high density. It also ensures that there exists at least one glyph with a low importance threshold in each region, thereby avoiding pockets of low density.

Visual Effects

As stated before, the glyph pool helps maintain temporal coherence by fixing glyph positions and avoiding glyph movement during visualization. To further improve temporal coherence when interacting with glyph-based visualizations, glyphs are gradually faded in and out of the scene by modulating the opacity and width of the glyphs. This provides a smooth transition and minimizes any popping artifacts caused when glyphs quickly toggle visibility when changing the importance field.

Streamline Visualization

Streamlines use lines to visualize the path a particle might take if it was released into a vector field and its direction and magnitude were influenced by the vector values it encountered. The placement and density of the lines are determined by the initial placement of seed points, that is, the starting locations for the imaginary particle, and the number of paths generated from seed points.

Choosing Seed Points and Streamline Density

Previous work [TB96, JL97] presents specialized methods for placing streamlines at a uniform density, but these methods present temporal coherence issues when changing the density. For example, Jobard and Lefer's method of image-guided seed point generation lead to bad temporal coherence because the next seed point is based on the previous streamline. If one streamline changes, the positions of all the streamlines may change. We use the randomized seed pool algorithm for seed point placement to better maintain temporal coherence and reduce distracting visual artifacts. This can been seen in Figure 7 in which overlays of two visualizations with slightly different focus point positions are shown. On the left, the results using image guided seed point positions and on the right, randomized seed pool. Notice how the image guide method causes changes in streamlines far from the focus point, while the seed pool method only results in changes to streamlines near the focus point.

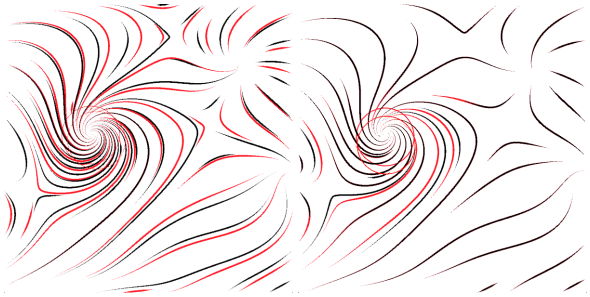


Figure 7. *Overlapping visualizations with slightly different focus point positions. Left, image guided seed placement results in almost every streamline changing. Right, the random seed pool results in almost all streamlines further from the focus point to remain constant.*

The randomized seed pool algorithm for seed point placement is better for maintaining temporal coherence and reduce distracting visual artifacts. A large number of candidate seed points are generated with a uniform distribution and high density. Each seed point is used to generated a streamline. If there is no valid streamline available (because there is already a streamline within a distance threshold of this seed point or because the streamline would be too short), the seed point is skipped. This process continues for all seed points. The more seed points that are chosen, the more likely that a seed point will be closer to an 'ideal' or 'optimized' seed point. The density of the visualization is also bound by the density of seed points. If the minimum separation distance d_{min} is small, then the number of seed points in the pool must be high or that distance cannot be achieved.

Streamline density is controlled by maintaining a separation distance d between streamlines. This separation distance d is defined as $d(x,y) = d_{max} - \mathbf{I}(x,y)(d_{max} - d_{min})$ and is used to determine if a seed point is valid and when to terminate integration of a streamline. The visualization space is divided into a uniform grid with each cell equal to some value between d_{min} and d_{max}. Each cell of this grid contains a list of points from other streamlines that lie within that cell. A point query only needs to test against all other points in the same cell and nearby cells; however, unlike Jobard and Lefer's method [JL97], all nearby cells within the search distance must be checked since the density is not uniform over the entire image.

Visual Effects

Streamlines are rendered as triangles instead of lines to allow for tapering and width variation along the streamline. Since the streamlines are oriented along one path, a single triangle strip can be used to render the streamline by starting at one end and alternating points between the left and right side of the streamline until reaching the other end. By using a triangle strip instead of several triangles, sending duplicate points to the GPU can be avoided.

This optimization can be taken a step further. If two "collapsed" triangles are added such that the end points are equal to the end of one streamline and the beginning of another, all the streamlines in the entire visualization can be rendered as a single triangle strip (with the connecting triangles having 0 area and being invisible). This optimization allows sending all the streamline data to the GPU at once rather than in packets.

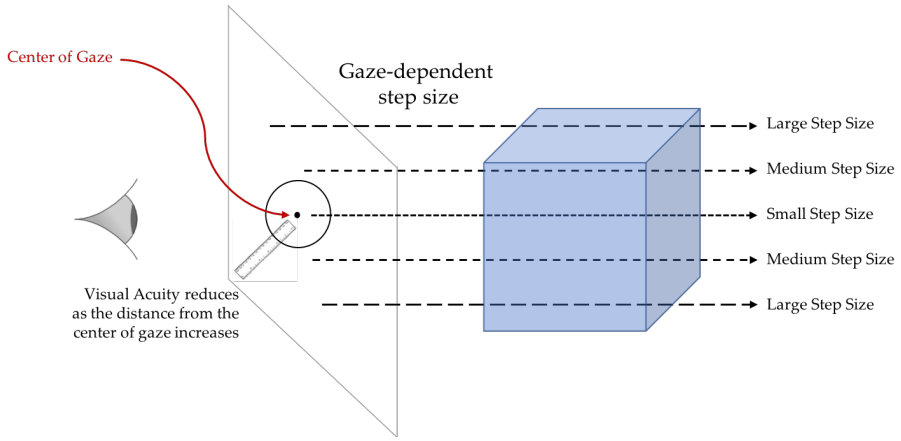


Figure 8. *This illustration explains our approach to foveated visualization of volumetric data. The region around the identified center point of gaze is where the step size for the rays cast into the scene have the lowest step size. We model the reduced visual acuity away from the center point of gaze by smoothly transitioning the step size to medium step size and large step size.*

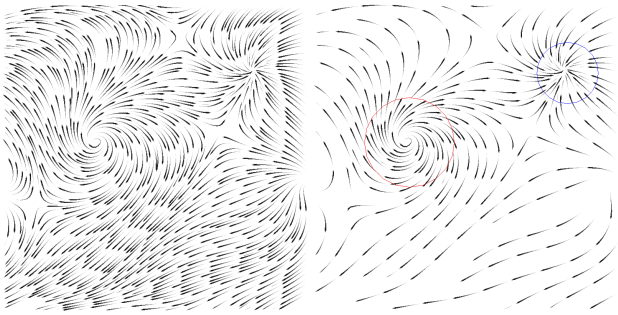


Figure 9. *Our streamlines with directional cues (left) and directed streamlines with foveated rendering (right) that eliminates visual clutter.*

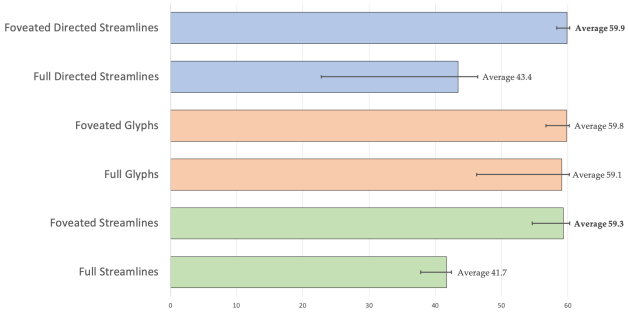


Figure 10. *Performance Analysis of Foveated Rendering of Vector data. The graph shows that the frame rates obtained for foveated rendering of streamlines and directed streamlines were consistently faster than full resolution rendering of streamlines and directed streamlines. For glyph rendering, the foveated rendering was faster as well but the full resolution rendering was comparably fast.*

To create *directed streamlines*, opacity and thickness can modulated by using a biased saw-tooth function. This makes the streamline appear as a series of streaks, similar to brush strokes, that conveys direction to the user. Figure 9 depicts our foveated rendering with directed streamlines.

Performance Analysis

We conducted a performance evaluation on a MacBook Pro (13-inch, 2017) running Mac OS 10.14.6 with an Intel Core i7 3.5 GHz processor with 16 GB RAM and an Intel Iris Plus Graphics 650 graphics processor. The data was collected over multiple datasets with ongoing interaction. As can be seen in Figure 10, the foveated versions of directed and regular streamlines are considerably faster than the full resolution versions. The foveated version of the glyph visualization technique is fast too, but the full resolution version is comparably fast. Our full resolution glyph rendering algorithm has been considerably optimized to reduce overlap and speed up rendering and so even though it shows all the data in the full resolution version, it is quite fast. Table 1 shows the minimum, average, and maximum framerates observed for these techniques.

Technique	Full Resolution			Foveated		
	Min	Avg	Max	Min	Avg	Max
G	47.3	59.1	60.2	57.1	59.8	60.2
S	38.5	41.7	42.4	55.6	59.3	60.3
DS	25.8	43.4	46.4	58.7	59.9	60.3

Performance analysis of the techniques in terms of the frames per second. G = Glyphs, S = Streamlines, and DS = Directed Streamlines. The Foveated Streamlines and Directed Streamlines perform much better than the full resolution versions. The foveated glyph visualization technique performs well but so does the full resolution glyph visualization technique.

Volumetric Data Visualization

Raycasting is a common method for realistically visualizing volumetric data and is particularly computationally intensive. The idea behind raycasting is that it simulates a ray of light from a light source, through a scene, and to the viewer's eye. As the ray moves through the scene, it reflects and refracts off objects and these bounces eventually contribute to the what the eye sees. The algorithm reverses the calculation by starting from a view point, projecting into each pixel in an image, and making its way through the scene to a virtual light source. Lower bounds on computation time depend on the desired visual effects such as

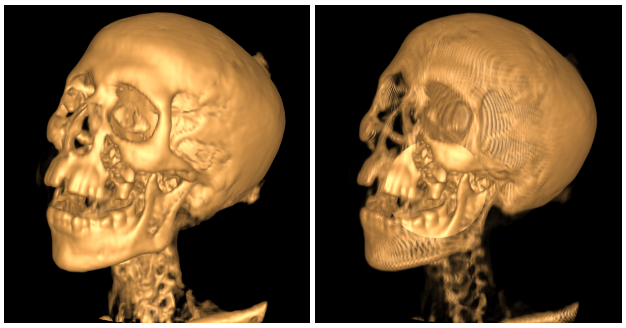


Figure 11. *Foveated visualization can cause hard transitions as shown in the figure on the right near the jaw of the mummy.*

shadows or reflections, and the complexity of the scene. However, the computation can be sped up by increasing the step size, or how fast a ray travels through the scene. Computationally, the step size refers to how much of a volume the ray moves in one iteration of the light equation, ranging from sub-voxel to multi-voxel steps. The step size directly impacts the computation time, but also impacts the quality of the image. As shown in Figures 1 and 12, the difference in image quality is caused by changing the step size.

In our foveated approach, we use the center of the user's gaze to adaptively modify the step size along the rays cast into the scene. As the distance from the center of the gaze increases radially outwards, we increase the step size in the raycasting algorithm. This in turn leads to the faster generation of the raycasted image and reduces interaction latency. Figure 8 shows a schematic that describes our foveated approach. By changing the step size of the raycasting algorithm based on distance to the central gaze point, the user can fluidly explore larger volumes and computation is dedicated most intensely areas being scrutinized by the user.

Image Quality Transitions

The goal of foveated visualization is to provide a seamless experience to the user. If we are not careful about transitioning smoothly between the region around the center of the gaze, the user will see an abrupt transition such as that shown in Figure 11 around the jaw.

We explore the various ways in which we can transition from the region of interest to the peripheral parts of the raycasted image. We explore six different transitions: Linear, Sharp Cut, Medium Cut, Smooth Cut, Slow Start, and Fast Start. We provide details about the various transitions below. Figure 12(a) shows high quality rendering with a constant small step size for all the rays that leads to lower frame rates for interaction. On the other hand, Figure 12(b) shows low quality rendering with raycasting with a higher step size for all rays that leads to better frame rates for interaction and low latency for users. The rest of the images in Figure 12 show the various transitions in comparison with the high-quality and low-quality (interactive) rendering options.

Linear Transition: For this transition from the region of interest outward, the step size starts with the smallest step size and linearly grows outward as the distance from the center point of gaze increases. Equation below shows how the step size is calculated for every ray with `Min` being the smallest step size, `Max` being the largest step size, and `Distance` being the *normalized* distance from the center point of gaze. Figure 12(f) shows the linear transition.

$$StepSize = Min + Distance * (Max - Min)$$

Sharp Cut Transition: In this transition, the step size is small for high quality rendering in the region of interest and high for the rest of the rays outside a user-controllable radius around the center of the gaze. For all our experiments, we set the normalized radius around the center of the gaze to be 0.2. This transition leads to a clear visual distinction between the region of interest and the rest of the rendered image that is not desirable in foveated visualization. Figure 12(c) shows the sharp cut transition with a clear visual difference between the region of interest around the center point of gaze and the surrounding area.

$StepSize = Min$
if $Distance > Radius$ **then**
  $StepSize = Max$
else
  $StepSize = Min$
end if

Medium Cut Transition: In this transition, the step size is small around the center point of gaze and slowly transitions out as the distance from the center increases. The improved visual quality difference is noticeable as compared to the Sharp Cut transition, but it comes at a very small performance hit of around 10% depending on the dataset being visualized. Figure 12(d) shows the medium cut transition with less of a visible difference between the region of interest and the surrounding region.

if $Distance < Radius$ **then**
  $StepSize = Min + (Distance)^2 * (Max - Min)$
else
  $StepSize = Min + \sqrt{Distance} * (Max - Min)$
end if

Smooth Cut Transition: This is the smoothest transition from the center of the gaze outward. The step size is small inside the region of interest and slowly transitions to the maximum step size as the distance increases outwards from the region of interest. The visual quality obtained is the best with a frame rate that is comparable to linear transition. Figure 12(e) shows the smooth cut transition with high quality rendering similar to Figure 12(a) with a 2.4x speedup in performance as compared to the high quality version.

if $Distance < Radius$ **then**
  $StepSize = Min + (Distance)^2 * (Max - Min)$
else
  $StepSize = Min + (Radius)^2 * (Max - Min) + \sqrt{Distance - Radius} * (Max - Min)$
end if

Slow Start Transition: This transition starts with a lower step size initially and slowly increases it as the distance from the center of the gaze increases. It leads to better visual quality than some of the other transitions. The equation below provides the details of the transition to compute the step size for each individual ray being cast. Figure 12(h) shows the slow start transition that leads to higher quality visualization as compared to fast start (Figure 12(g)) but it also leads to a slow down in performance.

$$StepSize = Min + (Distance)^2 * (Max - Min)$$

Fast Start Transition: This transition increases the step size

quickly as the distance from the center of the gaze increases. This leads to faster interaction rates with a noticeable drop in visual quality as compared to the Slow Start transition. Figure 12(g) shows the fast start transition that provides relatively lower quality rendering as compared to Smooth Cut (Figure 12(e)) or Slow start (Figure 12(h)).

$$StepSize = Min + \sqrt{Distance} * (Max - Min)$$

Performance Analysis

Technique	Framerate
High quality rendering	*8.68 fps*
Low quality rendering	*39.42 fps*
Foveated with Sharp Cut transition	*31.31 fps*
Foveated with Medium Cut transition	*24.64 fps*
Foveated with **Smooth Cut** transition	**20.94 fps**
Foveated with Linear transition	*21.19 fps*
Foveated with Fast Start transition	*31.12 fps*
Foveated with Slow Start transition	*15.56 fps*

Framerates for volume visualization transition types. As can be seen from Figure 12, Smooth Cut has the best tradeoff in terms of high quality and interactive frame rates.

We conducted experiments on multiple volumetric datasets. The application window size was set to 1000 * 1000 pixels. For foveated rendering, the radius was set to 200 pixels (0.2). Based on the gaze tracking, we measured the frame rate for each of the variations in transitions introduced before. The supplementary material has the detailed performance analysis for the various datasets used in our experiments.

If we focus on high quality rendering, Slow Start would be the best choice. The average improvement in frame rate compare to minimum constant step size is 200%. If we focus on high frame rate, Linear and Fast Start are good as they have 250% and 300% frame rate improvement on average. If we want the foveated region to stand out to draw the viewer's attention to specific regions, Medium Cut and Sharp Cut work well with 280% and 400% frame rate improvement on average. It is particularly hard to compare and select the *best* technique but the Smooth Cut has the best tradeoff in terms of high quality and interactive frame rates.

In general we found that the frame rates were lowest for Slow Start, followed by Smooth Cut, Linear, Medium Cut, Fast Start, and Sharp Cut. Sharp Cut has the highest frame rate for foveated rendering due to the nature in which the step size is computed for the entire image.

Conclusion and Future Work

We have shown in this paper that incorporating foveated imaging into a rendering system can greatly improve performance without noticeably impacting visual quality. Our techniques have targeted both vector and volume visualizations, both of which have technical considerations that needed to be integrated into this new system.

Future work for this system includes extending to eye-tracking approaches by identifying algorithms to stabilize the stimulus during periods of visual fixation and providing user control over parameters of the oculomotor events. We would also like to conduct a user study to determine factors such as parameters in our algorithms when users notice the transitions between high and low resolutions and develop guidelines for step-sizes and wave functions to weight the computational costs of those decisions with the perceptual detection of change.

References

[BG05] BRUCKNER S., GROELLER M. E.: Volumeshop: An interactive system for direct volume illustration. In *Proceedings of IEEE Visualization 2005* (Oct. 2005), C. T. Silva E. Groeller H. R., (Ed.), pp. 671–678.

[BG07] BRUCKNER S., GRÖLLER E.: Enhancing depth-perception with flexible volumetric halos. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (2007), 1344–1351.

[BGKG06] BRUCKNER S., GRIMM S., KANITSAR A., GROLLER M. E.: Illustrative context-preserving exploration of volume data. *IEEE Transactions on Visualization and Computer Graphics 12*, 6 (2006), 1559–1569.

[BK00] BRADSKI G., KAEHLER A.: Opencv. *Dr. Dobb's journal of software tools 3* (2000).

[BK08] BRADSKI G., KAEHLER A.: *Learning OpenCV: Computer vision with the OpenCV library*. "O'Reilly Media, Inc.", 2008.

[CL93] CABRAL B., LEEDOM L.: Imaging vector fields using line integral convolution. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques* (1993), ACM, pp. 263–270.

[CY97] CHANG E.-C., YAP C. K.: A wavelet approach to foveating images. In *Proceedings of the thirteenth annual symposium on Computational geometry* (1997), ACM, pp. 397–399.

[CYY*11] CHEN C., YAN S., YU H., MAX N., MA K.: An illustrative visualization framework for 3d vector fields. In *Computer Graphics Forum* (2011), vol. 30(7), Wiley Online Library, pp. 1941–1951.

[DCH88] DREBIN R. A., CARPENTER L., HANRAHAN P.: Volume rendering. *SIGGRAPH Comput. Graph. 22*, 4 (June 1988), 65–74.

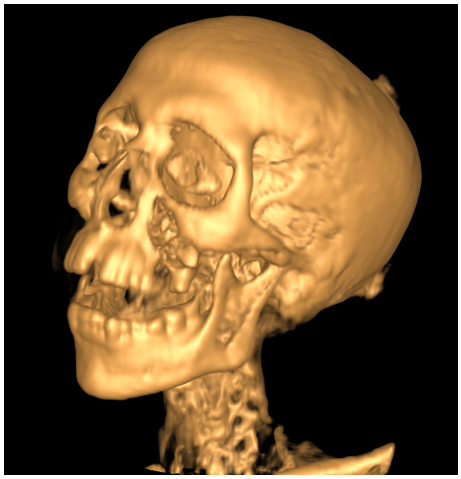
[GFD*12] GUENTER B., FINCH M., DRUCKER S., TAN D., SNYDER J.: Foveated 3d graphics. *ACM Transactions on Graphics (TOG) 31*, 6 (2012), 164.

[GNBP11] GASTEIGER R., NEUGEBAUER M., BEUING O., PREIM B.: The flowlens: A focus-and-context visualization approach for exploration of blood flow in cerebral aneurysms. *IEEE Transactions on Visualization and Computer Graphics 17*, 12 (2011), 2183–2192.

[GP98] GEISLER W. S., PERRY J. S.: Real-time foveated multiresolution system for low-bandwidth video communication. In *Human vision and electronic imaging* (1998), vol. 3299, pp. 294–305.

[HWHJ99] HECKEL B., WEBER G., HAMANN B., JOY K.: Construction of vector field hierarchies. In *Proceedings of the conference on Visualization'99: celebrating ten years* (1999), IEEE Computer Society Press, pp. 19–25.

[JL97] JOBARD B., LEFER W.: Creating evenly-spaced streamlines of arbitrary density. *Visualization in Sci-*

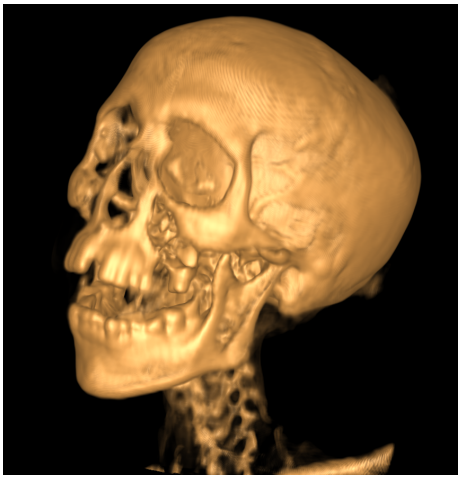


(a) High quality rendering *(8.68 fps)*

(b) Low quality rendering *(39.42 fps)*

(c) Foveated rendering with Sharp Cut transition *(31.31 fps)*

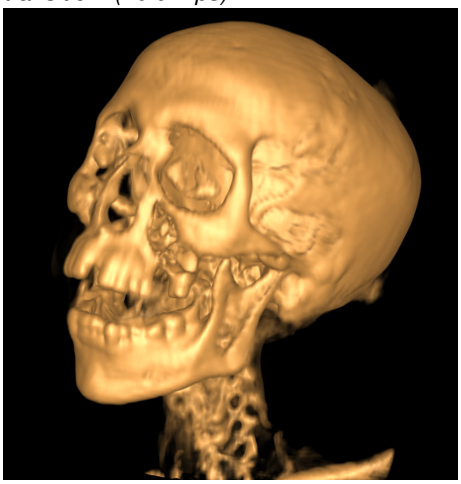
(d) Foveated rendering with Medium Cut transition *(24.64 fps)*



(e) Foveated rendering with Smooth Cut transition *(20.94 fps)*

(f) Foveated rendering with Linear transition *(21.19 fps)*

(g) Foveated rendering with Fast Start transition *(31.12 fps)*



(h) Foveated rendering with Slow Start transition *(15.56 fps)*

Figure 12. *Comparative rendering of the mummy dataset. Figure (a) shows high-quality rendering with a step size of 0.2. Figure (b) shows speed-focuse rendering with a higher step size of 1.0. Figure (c) shows the result of foveated rendering that includes high-quality rendering in the region of interest and lower-quality rendering for the rest of the image. Figures (d-h) show variations of foveated rendering.*

entific Computing 97 (1997), 43–55.

[Joh13] JOHNSON J.: *Designing with the mind in mind: simple guide to understanding user interface design guidelines*. Elsevier, 2013.

[KMS07] KALKOFEN D., MENDEZ E., SCHMALSTIEG D.: Interactive focus and context visualization for augmented reality. In *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality* (2007), IEEE Computer Society, pp. 1–10.

[LH08] LIVINGSTONE M., HUBEL D.: *Vision and Art: The Biology of Seeing*. Abrams, 2008.

[LKJ*05] LAIDLAW D., KIRBY R., JACKSON C., DAVIDSON J., MILLER T., DA SILVA M., WARREN W., TARR M.: Comparing 2d vector field visualization methods: A user study. *Visualization and Computer Graphics, IEEE Transactions on 11*, 1 (2005), 59–70.

[LRA*07] LI W., RITTER L., AGRAWALA M., CURLESS B., SALESIN D.: Interactive cutaway illustrations of complex 3d models. In *ACM Transactions on Graphics (TOG)* (2007), vol. 26(3), ACM, p. 31.

[PG02] PERRY J. S., GEISLER W. S.: Gaze-contingent real-time simulation of arbitrary visual fields. In *Human vision and electronic imaging* (2002), vol. 57.

[PSK*16] PATNEY A., SALVI M., KIM J., KAPLANYAN A., WYMAN C., BENTY N., LUEBKE D., LEFOHN A.: Towards foveated rendering for gaze-tracked virtual reality. *ACM Transactions on Graphics (TOG) 35*, 6 (2016), 179.

[RE01] RHEINGANS P., EBERT D.: Volume illustration: Nonphotorealistic rendering of volume models. *IEEE Transactions on Visualization and Computer Graphics 7*, 3 (2001), 253–264.

[RLMS03] REINGOLD E. M., LOSCHKY L. C., MCCONKIE G. W., STAMPE D. M.: Gaze-contingent multiresolutional displays: An integrative review. *Human factors 45*, 2 (2003), 307–328.

[SCK10] SCHROEDER D., COFFEY D., KEEFE D.: Drawing with the flow: a sketch-based interface for illustrative visualization of 2d vector fields. In *Proceedings of the Seventh Sketch-Based Interfaces and Modeling Symposium* (2010), Eurographics Association, pp. 49–56.

[SJM96] SHEN H., JOHNSON C., MA K.: Visualizing vector fields using line integral convolution and dye advection. In *Volume Visualization, 1996. Proceedings., 1996 Symposium on* (1996), IEEE, pp. 63–70.

[SRIR07] SANTINI F., REDNER G., IOVIN R., RUCCI M.: Eyeris: a general-purpose system for eye-movement-contingent display control. *Behavior Research Methods 39*, 3 (2007), 350–364.

[TB96] TURK G., BANKS D.: Image-guided streamline placement. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (1996), ACM, pp. 453–460.

[TGK*17] TOMINSKI C., GLADISCH S., KISTER U., DACHSELT R., SCHUMANN H.: Interactive lenses for visualization: An extended survey. In *Computer Graphics Forum* (2017), vol. 36(6), Wiley Online Library, pp. 173–200.

[TVW99] TELEA A., VAN WIJK J.: Simplified representation of vector fields. In *Visualization'99. Proceedings* (1999), IEEE, pp. 35–507.

[TWSK14] TAO J., WANG C., SHENE C.-K., KIM S. H.: A deformation framework for focus+ context flow visualization. *IEEE transactions on visualization and computer graphics 20*, 1 (2014), 42–55.

[UFH12] UŘIČÁŘ M., FRANC V., HLAVÁČ V.: Detector of facial landmarks learned by the structured output svm. *VIsAPP 12* (2012), 547–556.

[VKG05] VIOLA I., KANITSAR A., GROLLER M.: Importance-driven feature enhancement in volume visualization. *Visualization and Computer Graphics, IEEE Transactions on 11*, 4 (2005), 408–418.

[WS01] WISCHGOLL T., SCHEUERMANN G.: Detection and visualization of closed streamlines in planar flows. *Visualization and Computer Graphics, IEEE Transactions on 7*, 2 (2001), 165–172.

[WSF*95] WITTENBRINK C., SAXON E., FURMAN J., PANG A., LODHA S., ALPER N., FERNANDEZ D., KOLSKY H., NUSS W.: Glyphs for visualizing uncertainty in environmental vector fields. In *SPIE & IS&T Conference Proceedings on Electronic Imaging: Visual Data Exploration and Analysis* (1995), Citeseer, pp. 87–100.

[WYM08] WANG C., YU H., MA K.: Importance-driven time-varying data visualization. *Visualization and Computer Graphics, IEEE Transactions on 14*, 6 (2008), 1547–1554.

[YCHZ05] YU H., CHANG E.-C., HUANG Z., ZHENG Z.: Fast rendering of foveated volumes in wavelet-based representation. *The Visual Computer 21*, 8-10 (2005), 735–744.

[ZSH96] ZOCKLER M., STALLING D., HEGE H.: Interactive visualization of 3d-vector fields using illuminated stream lines. In *Visualization'96. Proceedings.* (1996), IEEE, pp. 107–113.