View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Publications at Bielefeld University

Model-based Real-time Visualization of Realistic Three-Dimensional Heat Maps for Mobile Eye Tracking and Eye Tracking in Virtual Reality

Thies Pfeiffer*, Cem Memili†
Center of Excellence Cognitive Interaction Technology, Bielefeld University, Bielefeld, Germany

Abstract

Heat maps, or more generally, attention maps or saliency maps are an often used technique to visualize eye-tracking data. With heat maps qualitative information about visual processing can be easily visualized and communicated between experts and laymen. They are thus a versatile tool for many disciplines, in particular for usability engineering, and are often used to get a first overview about recorded eye-tracking data.

Today, heat maps are typically generated for 2D stimuli that have been presented on a computer display. In such cases the mapping of overt visual attention on the stimulus is rather straight forward and the process is well understood. However, when turning towards mobile eye tracking and eye tracking in 3D virtual environments, the case is much more complicated.

In the first part of the paper, we discuss several challenges that have to be considered in 3D environments, such as changing perspectives, multiple viewers, object occlusions, depth of fixations, or dynamically moving objects. In the second part, we present an approach for the generation of 3D heat maps addressing the above mentioned issues while working in real-time. Our visualizations provide high-quality output for multi-perspective eye-tracking recordings of visual attention in 3D environments.

Keywords: 3D, eye tracking, heat map, visualization

Concepts: •**Human-centered computing** → **Heat maps;** •**Computing methodologies** → **Perception;**

1 Introduction

The availability of mobile eye-tracking devices has fostered an increased interest in investigating visual attention in life-sized realistic environments. Mobile eye tracking, however, is more difficult and resource-intensive to analyse than desktop-based eye tracking. In particular, current visualization techniques for desktop-based eye tracking do not directly transfer to mobile scenarios, due to problems such as dynamic perspectives, occlusions or dynamic environments. These issues will be discussed in detail in Section 2.

Our work aims at creating realistic 3D heat maps for dynamic 3D environments, subsuming natural environments recorded with mobile eye-tracking devices as well as eye-tracking recordings in virtual reality. As a prerequisite, we assume that a 3D model of the environment is available. This model can either be created manually or using techniques such as 3D scanning. For dynamic environments the approach also assumes that changes in the environment can be tracked. While this assumption will currently not hold for many application cases of mobile eye tracking, research in other areas, such as 3D reconstruction, will open up more application areas in the future.

Figure 1: *The presented approach creates realistic heat maps for objects that have been moved around and inspected from all sides in real-time with a high-quality. Here the example of a product picked from a shelf of a virtual supermarket (see also Figure 5).*

We have successfully applied the presented approach to situations in which participants inspect a static real-world scenario, such as a life-sized car or a kitchen, and to eye-tracking studies in interactive virtual reality simulations, such as a virtual supermarket, in which the dynamic changes of all objects are known at a very high precision (see Figure 1). The approach integrates nicely with our work on EyeSee3D for real-time tracking of visual attention in 3D environments based on mobile eye-tracking devices presented also in these proceedings [Pfeiffer et al. 2016].

The presented approach is unique in that it takes into account the depth of fixation, occlusions of foreground objects, binocular eye tracking and moving and deforming objects. Furthermore, on the more technical side, it uses multi-texturing to visualize the heat maps on the target objects' surfaces, defers the costly normalization process to a real-time shader, and outputs a set of persistent heat map textures that can be used to create high-end photo-realistic renderings offline (e.g. for documentation). The approach is optimized to work on GPUs and operates in real-time for compact scenarios.

The structure of the paper is organized as follows: in the next section, we start with an analysis of the requirements and challenges of heat mapping that are specific for dynamic 3D environments. Once the grounds are layed, we review the related work in light of our anaylsis in Section 3, before we present our own approach in Section 4. In Section 5, we describe the technical implementation and provide details relevant for reproducing the work, but also for understanding hardware-based limitations. We then pick up important technical challenges and address them in Section 6, before we conclude in Section 7.

*e-mail: thies.pfeiffer@uni-bielefeld.de
†e-mail: cmemili@techfak.uni-bielefeld.de
Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org. © 2016 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ETRA 16, March 14 - 17, 2016, Charleston, SC, USA
ISBN: 978-1-4503-4125-7/16/03
DOI: http://dx.doi.org/10.1145/2857491.2857541

2 Analysis of Heat Mapping for Dynamic 3D Environments

Before we enter into the analysis, we want to distinguish between two roles: With *viewer* we will from now on refer to someone whose gaze is recorded during the inspection of a scene. A *reviewer*, in contrast, is someone investigating the recorded attentional data. The gaze of the *reviewer* is not necessarily recorded.

Several disciplines have interests in representing and visualizing areas that attract the visual attention of viewers and historically different terms have been introduced to refer to rather similar things: attentional landscape [Pomplun et al. 1996], saliency maps, attention maps, heat map (which is a more general term but often used interchangeably with attention map, e.g., in the domain of human factors/user experience) or recently 3D attentional maps or 3D attention volumes. In the following we will use the term heat map.

Starting point of our discussion are situations in which the spread of visual attention over 3D stimuli, either real or virtual, is to be recorded and visualized via a heat map. For the recordings, the viewer might either wear a mobile eye-tracking device and walk around in a physical setup, sit in front of a desktop virtual-reality system equipped with a remote eye-tracking system or be equipped with a mobile eye-tracking device embedded in a head-mounted display or in the 3D glasses used for a Powerwall or CAVE.

2.1 Requirements

A correct heat map visualization for dynamic 3D environments has to address the following requirements:

Estimating the 3D point of regard: When tracking gaze on 2D stimuli, only X and Y coordinates are relevant. Mobile eye-tracking systems already need to correct for parallax shifts introduced by the different levels of depth of the objects recorded in the scene camera video. For a realistic analysis of visual attention in 3D, however, a correct estimation of the level of depth of a fixation is required to determine the 3D point of regard.

Modelling 3D attention spreading: A correct model for representing the spread of attention around the measured 3D point of regard has to be implemented. In the eye-tracking literature it is common to assume a 2D Gaussian distribution for that (see formula (1) later in this paper).

Handling of (partial) occulsions: Objects distributed over a scene might (partially) occlude each other depending on the perspective of the viewer. Occlusions may then continuously change over time as the viewer or the objects are moving.

Handling of dynamic objects: As soon as gaze data is aggregated over time, dynamic objects can become problematic. Most approaches for generating heat maps are using the global 2D screen space for representation. However, as soon as an object moves during the observed timespan, the attention attributed to the object's location will stick to the screen space previously occupied by the object. This results in invalid visualizations. For realistic heat map visualizations, the representation has to be attached to the object.

Handling of multiple viewers/perspectives: In desktop-based 2D situations, the perspective of the viewers is typically fixed and all viewers perceive exactly the same 2D image. In 3D environments and in particular in such of natural sizes, viewers perceive the scene from different perspectives due to the mere facts that they are of different heights and they are moving around.

Figure 2: *Example workflow: inspection of a 3D CAD prototype of a car in a virtual reality environment (right), real-time generation of attention textures (top left) and high-quality rendering of the aggregated heat maps (lower left).*

2.2 Technical Challenges

In addition to the substantial requirements, the following technical challenges are in our opinion relevant to be tackled:

Real-time availability: The aggregation of gaze data and the computation of the heat map is a computationally demanding process. Creating heat maps as quickly as possible is not only relevant for an on-line monitoring of gaze data, e.g. for quality control of recordings, it will also decrease the time needed for generating visualizations for offline review.

Easy dissemination: Heat maps are used for qualitative communication between researchers and their audience. It is therefore crucial that the audience can access the visualizations in the most simple way possible. Recent approaches for generating heat maps for 3D scenes (e.g. [Maurus et al. 2014]), however, require special purpose viewers and dedicated graphics hardware. Besides the problem of distributing this software, such tools also increase the learning curve for the audience.

Integration into existing workflows: Especially when reviewing designs, it is important that all visualizations are of high-end quality. This can be achieved best if the results can be re-integrated into the existing workflow (CAD, CAM) to create appropriate renderings (see Figure 2).

Some self-explanatory requirements are the handling of arbitrary scenes, a low manual effort in preparing and setting up the analysis and the support of multiple reviewers.

3 Related Work

A general overview about the state of the art in visualizing eye tracking data, which also covers heat maps for 2D images, videos and 3D objects is provided by Blascheck et al. [2014]. Here we focus on the discussion of heat map visualizations for scenes for which 3D models are available. The issue of correctly creating heat maps for 3D environments has, as far as we know, only been considered by a small number of researchers. We thus decided to provide a more in-depth discussion of the few individual contributions to work out the differences of the presented approach to the body of previous methods, instead of providing a broader coverage of the field of heat maps for 2D stimuli. We are also postponing the discussion of whether or not a realistic analysis of a 3D environment has advantages or disadvantages over the reduction of the stimuli to 2D to subsequent research. However, in our opinion visualizations of the spead of visual attention during interactions with ob-

jects, such as depicted in Figure 6, provide a strong argument for following this line of research.

3.1 Stellmach et al.

Several approaches to visualize visual attention on static virtual 3D scenes have been described by Stellmach et al. [2010]. Their *Projected Attentional Maps* are created based on recorded 3D fixations. They are 2D heat maps which are overlaid over the 3D scene visualization according to the current perspective of the reviewer. In that they are close to standard 2D heat maps. However, they ignore the 3D structure of the scene. For their *Object-based Attentional Maps* Stellmach et al. aggregate fixations on a per-object level and colorize the object with a uni-colored texture accordingly. Their last approach, the *Surface-based Attentional Maps*, breaks the attention analysis down to the level of triangles defining the object's geometry. Each triangle (or vertex) of the object is assigned a color representing the aggregated and normalized attention. Thus Stellmach et al. create a "second skin", as they call it, to represent the Surface-based Attentional Map. This approach depends, as the authors note, heavily on the granularity of the mesh. For a fine-grained analysis a high-density mesh is required. This geometry-based approach has a severe negative affect on visualization performance.

Note also that the latter two approaches are only considering the first object that has been hit by a fixation. Attention that might have been spread to other objects in close vicinity is ignored. While the authors not explicitly mention dynamic scenes, their two object-centered approaches (Object-based Attentional Maps and Surface-based Attentional Maps) could also be applied to dynamic objects. An advantage of the Object-based Attentional Map is that the resulting colorizations could be exported and used in a high-end rendering system, if the coarse object-level granularity is not of importance. This approach is thus not useful for design reviews: In their user study, Stellmach et al. [2010] could show that the granularity achieved by the Surface-based Attentional Maps was rated more useful than the simple Object-based Attentional Maps. While the heat map mesh of the Surface-based Attentional Map could also be exported, the re-use in a high-end rendering system is not as simple as with texture-based approaches.

A major problem that remains with the work presented by Stellmach et al. [2010] is the way the 3D fixations have been collected in the first place. They only considered gaze alike mouse input and used picking to determine the first triangle hit by a ray cast into 2D screen space (the experiments used only monocular desktop-based virtual reality). They did not consider the depth of the fixation nor did they consider visibility when spreading the attention towards adjacent triangles.

3.2 Duchowski et al.

Duchowski et al. [2012] presented a GPU-based algorithm for the creation of 2D heat maps for visual attention data collected from several participants watching the same stimulus video. As their system operates on 2D material only, the perspectives of viewer and reviewer are fixed. The attention data is also not linked to the attended area presented in the video, but to the position fixated on screen. Any temporal aggregation thus results in incorrect mappings to the presented content if either the camera or the presented objects are moving. Their overall approach is (1) accumulating Gaussian distributions for each fixation, (2) searching for maximum intensity for normalization, (3) normalization, and (4) coloring according to a selected color ramp. All of these processes were computed on the GPU, for searching the maximum intensity, they used GPU parallel reductions [Buck and Purcell 2004]. The authors note that their approach for 2D attention maps could in principle be extended to support 3D scenes, which is what is elaborated in this paper.

3.3 Work in the ARtSENSE Project

In previous work in the European project ARtSENSE, Hammer et al. [2013] used a 3D model of the environment and outside-in tracking based on the scene-camera video of the eye-tracking system to determine 3D-fixation locations during the viewer's visit to a museum. Based on this work, they developed a GPU-accelerated visualization-system for heat maps [2014]. Their contribution to the state-of-the-art is that their approach uses a 3D scene and projects the viewers' gaze into the scene while correctly respecting occlusions imposed by objects closer to the viewer than the fixated object by using Shadow Mapping. To create a heat map from the perspective of the reviewer, they project a 2D Gaussian distribution for each considered fixation modeling the spread of attention from the perspective of the corresponding viewer into the scene and collect all these projections in a shader to create a real-time texture overlay for the screen space, similar to the Projected Attentional Maps but considering occlusions for the perspective of the reviewer. Their approach, however, has the drawback that a shadow map has to be calculated and stored in a texture buffer for every distinct viewing direction of the viewers, which in a mobile setting could be as many as sampled fixations. As a consequence, the authors suggested to use textures of a low resolution, but this results in degradation of the visual quality.

3.4 Own Previous Work

Previously, we presented work on 3D attention visualization for viewers moving in an immersive virtual reality environment [Pfeiffer 2012]. We also discussed means to estimate the 3D point of regard based on binocular eye-tracking and eye vergence. For visualization, we proposed 3D volume renderings which are represented independently of the scene geometry. While the 3D point of regards provide a sound basis for attention visualization and the attention is spread over nearby objects, the overall approach creates unrealistic heat maps, similar to the *Projected Attentional Maps*, as the scene geometry is not taken into account: it uses no visibility checks and attention spreads into the insides and beyond the backside of objects as well. Also, the approach is memory-wise and computationally demanding and are not subject to incremental real-time visualizations except for small scenes.

The approach presented in this paper extends the state-of-the-art in several ways. First, it is able to consider real 3D distributions around the 3D point of regard. Second, the 3D point of regard can be computed either by the geometric approach by intersecting the line of sight with the 3D model, e.g., when using a monocular eye-tracking system, or more realistically by making use of binocular eye-tracking and depth estimations based on eye vergence. Third, the aggregated attention data is not stored per view but per object. This results in several improvements compared to previous approaches: (a) whether visual attention is analyzed can be selected on a per-object level, e.g., if only certain exhibits in a museum are of interest for the analysis, (b) the required memory scales only with the number of objects and not with the number of participants or the duration of the observations, (c) the created heat map textures can be easily made persistent on a per-object basis and (d) the created textures can be used to integrate the heat maps in high-quality renderings of the whole scenery or individual artifacts. The resulting 3D model incorporating the heat map visualization can then be viewed in any 3D presentation software, a special purpose viewer is no longer required after data aggregation.

4 Proposed Approach: Attention Textures

The basic idea is to represent the 3D heat maps as additional textures on a per-object level and to generate these textures directly on the GPU. This texture will be referred to as *Attention Texture* in the following.

4.1 The Basic Algorithm

Our algorithm takes as input a set of viewing events E that are tuples of (head_pos, head_orient, left_eye_pos, right_eye_pos, left_viewing_direction, right_viewing_direction, viewing_duration). Instead of the viewing directions also a previously estimated 3D point of regard can be used. The heat maps are then generated according to the following procedure:

1. Loop over all viewing events in the consideration set
   (a) Handle object occlusions for the current viewing event (viewer's perspective)
   (b) Map the spread of attention around the 3D point of regard on the objects' Attention Textures
2. Compute the global maximum attention value over all Attention Textures
3. Render the heat mapped scene from the reviewer's perspective

The core ideas of the approach are the aggregation of attention in object-local Attention Textures, the use of up to two shadow maps for a correct treatment of occlusion, the 3D model for attention spreading and the defered creation of the heat map only for rendering the view and not for every added fixation event.

4.2 Handling Object Occlusions

First, the viewer's perspective is taken from the current viewing event. From this perspective, the algorithm computes a so-called shadow map (see Section 5.1) to determine partial occlusions up to the depth of the viewing event's point of regard. This procedure is similar to the approach used by Hammer et al. [2013], however, the technical implementation is different, as will be explained in Section 5.1.

A special treatment is required for the visualization of binocular eye-tracking data. To create more realistic heat maps, the algorithm computes a second shadow map, one for each eye of the viewer, and considers both shadow maps when determining visibility. If a 3D point is visible for at least one eye, the corresponding Attention Texture is updated. This is for example relevant to handle fences or lattice windows correctly.

4.3 Realistic Mapping of Attention

A significant difference to previous work is that we compute a real 3D Gaussian distribution instead of just projecting the 2D Gaussian distribution onto objects. Our approach can also easily be extended to support even more realistic 3D models of the spread of attention. However, the 3D Gaussian already creates more realistic visualizations, as it respects the depth of focus, as depicted in Figure 3. For example, when the viewer is focusing on a nearby object, attention is no longer falsely attributed to far away background objects, which appear only very blurry on the retina. When using the 3D Gaussian mapping, the parameters of the Gaussian have to be adapted to the depth of the point of regard as the area of similar accuracy increases with increasing distance (flashlight effect).

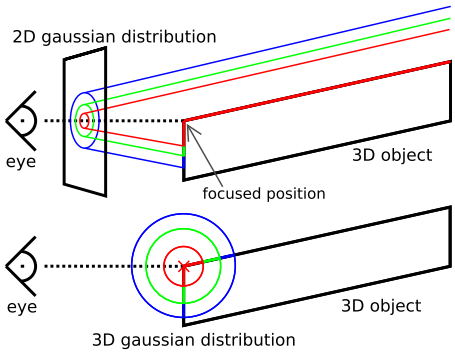

Figure 3: *The projected 2D Gaussian distribution used in previous approaches will create artifacts (upper picture). The 3D Gaussian mapping does not create these artifacts (lower picture) and is more realistic as it correctly considers the depth of fixation.*

Note that this behavior depends on the design of the mapping function, so if the previous behavior is still wanted, a cone-like Gaussian extending from the eye of the viewer towards and beyond the fixated target can be used to model the attention spread to achieve the previous visualizations.

Where the calculated 3D distribution intersects with an object, the corresponding position is checked against the current shadow map and if the position is visible to the viewer, the object's Attention Texture at that position is updated accordingly. Note that in contrast to the approach used by Stellmach et al. [2010], the Attention Texture's values are representing the absolute overall attention distribution; the heat map generation is defered to the rendering step. Figure 4 depicts the result of this mapping process for a single product of a virtual supermarket (see Figure 5).

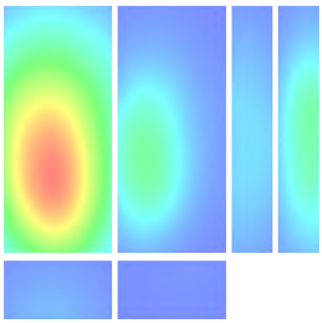

Figure 4: *This is an example of an Attention Texture for one of the products in a virtual supermarket. The texture has been colorized to make the different values light up according to the heat map color mapping. The individual boxes correspond to the different package sides.*

4.4 Computing the Global Maximum Attention Value

To visualize heat maps, the values in the Attention Textures have to be normalized to project them on the desired color ramp. Therefore, whenever Attention Textures have been updated and a heat map has to be drawn, the global maximum of all attention values has to be computed. Following the idea proposed by Duchowski et al. [2012] for the similar problem of determining the maximum value of a single heat map, a naive approach would have to iterate over all Attention Textures to determine the individual peaks and then, again, it would have to iterate over all peaks to find the global maximum.

Figure 5: *Example of a virtual supermarket scenario: In this life-sized immersive virtual reality installation participants could examine product packages in 3D.During these tasks, participant's eye movements were tracked using a mobile eye-tracking system.*

This would render the approach described so far intractable for real-time rendering with reasonable numbers of objects. Our efficient technical solution to this is described in Section 5.3.

4.5 Rendering for Reviewer or Export

Normalization and color mapping of the heat map is defered to rendering and efficiently computed on the GPU by shaders. This reduces the computational demands significantly, as otherwise all textures of all objects in the scene would have to be updated for each viewing event. With the optimized approach, only Attention Textures of visible objects within the range of the 3D distribution around the 3D point of regard have to be updated.

To augment the scene with the heat maps, a simplified Phong-Shading approach [Wolff 2011] is used with a colorization according to a selected color ramp (for heat maps typically rainbow colors are used, but see the discussion in [Duchowski et al. 2012] that this is perceptually misleading). An alpha value is used to blend the computed heat map color over the original color of the object (see Figure 1). For this approach, the linear filtering (GL_LINEAR) can be applied to achieve more appealing visual results.

One performance improvement of the presented approach is achieved by normalizing the attention values on the fly, thus saving expensive memory operations. As every object's Attention Texture would have to be updated, such an approach would quickly be intractable in real-time. However, we also implemented a shader that creates a new set of textures for the objects based on the Attention Textures and the chosen mapping. These textures can be exported to make the heat-mapped results persistent. The created heat map textures can be imported into the original model, either to explore the persistent attention data in the original simulation framework (e.g. using a CAVE or an HMD) or to create high-end renderings of the collected data. Figure 6 shows a rendering of the exported Attention Texture using the open-source software Blender.

Figure 6: *For each product in the shelves, 3D attention textures have been calculated and high-quality renderings have been created. As participants were able to pick the packages out of the shelf (see Figure 5), fixations were distributed all over the package (front, back, sides).*

5 Implementation

In the following, we describe more technical details about the heat map generation. Some of these details are specific to the currently available graphics hardware. More technical details, include the program, pictures and videos can be found on the accompanying website [Pfeiffer and Memili 2016].

5.1 Computing the Shadow Map

We use hardware shadow-mapping [Everitt et al. 2001] to compute occlusions for each viewing event. As detailed above, we want to correctly handle binocular viewing perspectives and thus have adopted the shadow mapping algorithm accordingly. So instead of a single rendering pass for creating the shadow map, we use two subsequent passes.

5.2 Shader-based Attention Spreading

Attention spread is computed by rendering all relevant objects using a dedicated shader. The Attention Texture is used both as render target for the active frame-buffer object and as sampler for the same shader to be able to retrieve previous values. We avoid concurrency problems by refraining from parallel accesses to the same pixels. Note that texture filtering has to be set to GL_NEAREST to avoid artifacts with surrounding texture pixels not belonging to the relevant region of the object. The appropriate vertex shader is printed in Listing 1.

```
1  vertex.texCoords = a_texCoord0;
2  gl_Position = vec4(a_texCoord0 *2 -(1.0,1.0)↩
       ,0.0,1.0);
3  vertex.position = u_projView * a_position;
4  vertex.normal = (u_projView * vec4(a_normal,1)).↩
       xyz;
5  vertex.lightVertexPositionLeft = shadowMatrixLeft↩
       * a_position;
6  if(numberOfEyes == 2) {
7    vertex.lightVertexPositionRight = ↩
       shadowMatrixRight * a_position; }
```

Listing 1: *Vertex shader to prepare the update of Attention Textures.*

The fragment shader is then straightforward (see Listing 2): it reads the previous value from the Attention Texture, checks if the 3D po-

sition is within a relevant range of the Gaussian distribution (lines 4-6), then checks the shadow maps if the point is visible (line 11) and only then computes the Gaussian value and updates the Attention Texture (line 13).

```
1  float AV = (texture2D(AT, texCoords)).r;
2  float maxV = texture2D(maxAT, texCoords).r;
3  //0. compute distance to gaussvolume
4  float dist = distance(point_of_regard, position.↩
       xyz);
5  //1. check if inside the Gauss distr.
6  if(dist > GaussRadius) {
7   //early out possible if not inside
8   exitNoUpdate(AV, maxV);
9  } else {
10  //2. check if visible for defined eyes
11  if(isVisibleForEyes()) {
12   // update attention texture
13   exitWithUpdate(dist, AV, maxV);
14  } else {
15   // fragment is occluded
16   exitNoUpdate(AV, maxV); }}}
```

Listing 2: *Fragment shader to update of Attention Textures.*

Occlusion is handled by *isVisibleForEyes()* based on the computed shadow maps. It also handles z-Fighting problems by using an epsilon of 0.001 during the tests. The model of the 3D attention spread, here the 3D Gaussian model (1), is finally applied for visible pixels. It is implemented in a user function called by *exitWithUpdate*.

$$A(\vec{x}) = d(t)e^{-\frac{|\vec{x}-\vec{p}_{POR}|^2}{\sigma(\vec{p}_{eye},\vec{x})}} \quad (1)$$

with

$\vec{p}_{POR}$: 3D point of regard

$\vec{p}_{eye}$: 3D position of the eye (left or right)

$d(t)$: time-dependent scaling

5.3 Speeding up Maximum Computation: the Max-Attention Texture

In a sequential approach to normalization, it is sufficient to store a single maximum value and update that whenever a higher value is computed. This, however, does not work when updating the attention values in parallel using the shader. An ad-hoc approach would determine the maximum attention value per texture during the update and then, in a post-processing step, find the maximum of all of these object-local maxima. This, however, has two problems: First, storage would be needed that scales with the number of attention textures and second, the shader would run into concurrency problems when several parallel updates of the current maximum value take place.

Therefore, our algorithm does not compute the maximum on a per object basis, but per texture coordinate. For this it only requires one additional texture the size of the largest Attention Texture being used. The fragment shader (Listing 2) checks in the *exitWithUpdate* function whether the current attention value exceeds the maximum value over all textures for that coordinate using the Max-Attention Texture and updates this texture if necessary. The maximum attention value can now be determined using the parallel reductions approach [Buck and Purcell 2004].

6 Technical Constraints and Optimizations

While the approach presented so far already provides real-time performance for medium-sized scenes, the fillrate needed for updating the Attention Textures can be a bottle-neck. A low precision of the shadow maps can lead to visual artifacts. We therefore discuss some options for optimizations.

6.1 Reducing the Fillrate

To reduce the fillrate, we added a standard viewing-frustum culling based on bounding spheres to narrow down the set of target objects whose Attention Textures need to be updated. We also use the effective size of the Gaussian distribution around the 3D point of regard as a second optimizing criterion. Note that we need all objects in between the viewing position and the 3D point of regard to correctly compute the shadow maps, so we cannot just cull down to the Gaussian.

Crowded areas with many smaller target objects could still degrade performance. One way to further reduce the number of tackled pixels is back-face culling, which would ideally cut memory access in half. As the standard OpenGL back-face culling is not compatible with our optimizations and the binocular occlusion detection, we implemented back-face culling in a custom geometry shader that is applied between the vertex and the fragment shader.

6.2 Increasing the Precision of the Shadow Maps

A typical problem of shadow maps is that the precision of the depth values is constrained by the restrictions of the OpenGL texture. If precision is too low, false positives will lead to unrealistic attention distributions. While we use OpenGL textures of a very high precision, the extension of the relevant scene between the near and the far plane still has a significant effect on the net precision that can be achieved. We optimize the near and far planes according to the depth of the objects closest to the viewer's position (near plane) and the depth of the 3D point of regard (far plane) plus the effective radius of the Gaussian distribution (see Figure 7).

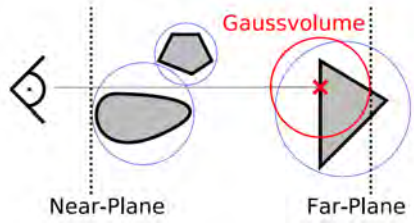

Figure 7: *Optimizing near and far planes for shadow mapping.*

The next possible improvement would be using Cascaded-Shadow-Mapping [Dimitrov 2007], Parallel-split Shadow Mapping [Zhang et al. 2006] or Sample Distribution Shadow Maps [Lauritzen et al. 2011], which improve upon the precision of the shadow maps if the presented approach is still not sufficient (e.g. with very large scenes and very distant points of interest).

6.3 Precision of the Attention Textures and Number of Supported Viewing Events

This leads us to the discussion of the accuracy and number of events that can be supported by the Attention Textures. Precision is bound to the format of the texture. On a single logical Attention Texture,

we use the single-channel high-precision format GL_R32F supported by todays consumer graphic cards, providing a 32bit floating point value.

The precision of the Attention Textures also limits the maximum number of viewing events. If many viewing events are targeted at the same pixel over a longer period of time, the limit of the texture's floating point representation can be reached. With the normal single-channel Attention Texture representation, the approach can account for about 1 million viewing events on the same pixel. After that, increasing numbers can still be represented, but with a loss of precision due to rounding effects. However, with an eye tracker running at 60 Hz, this problem raises only if viewers are staring for 4.6 hours at exactly the same pixel (given 8 digits of precision).

If this is actually a problem, then additional color channels could be used to improve the range of the representation. This would, however, mean that the shader has to do the maths to represent the floating point value. By combining two channels, viewers could then stare at the same position for about 53,000 years.

6.4 Optimal Resolution of Attention Textures and Texture Overlapping

An important factor determining the performance of the presented approach is the size of the Attention Textures. In our own tests we used textures sized 1024×1024 ($4.2\,MB$ or 237 objects per GB of memory). But if less accuracy is sufficient, decreasing the textures to 512×512 or 256×256 significantly reduces the fillrate and memory requirements, which can be used to either increase performance or analyse more target objects at the same time.

The reduction of the texture size, however, comes at a cost (see Figure 8): by reducing the size, the same number of texture coordinates needs to be mapped to less and less pixels. This increases the chance of overlapping mappings (red areas in Figure 8). It may even happen that certain areas of the object's surface might not be mapped to valid texture areas any more, which will result in holes in the visualized heat map.

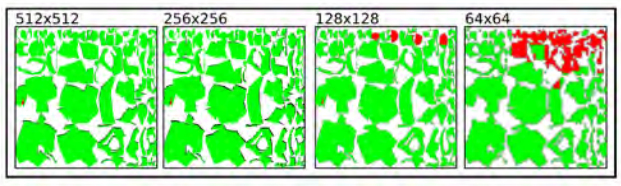

Figure 8: *If the size of an Attention Texture is decreased, this not only affects the accuracy, but also might induce mapping problems, highlighted here in red.*

Overlapping texture mappings are also found in manually created models: if several areas of an object show similar textures, the modeler might decide to optimize texture usage by mapping different regions of the object onto overlapping areas of the texture. While this may help to save memory for the original object, the original texture mapping coordinates can no longer be used for the creation of the Attention Textures. As Figure 9 shows, the attention values attributed to one area of the object would then also show up on every other surface area of the object that shared the same texture coordinates.

This problem can only be solved by remapping the texture coordinates so that they induce no overlays. This can be done either manually or with the support of state-of-the-art modeling tools. At the time being, our implementation checks for this problem and provides a warning, indicating the problematic areas (similar to Figure 8).

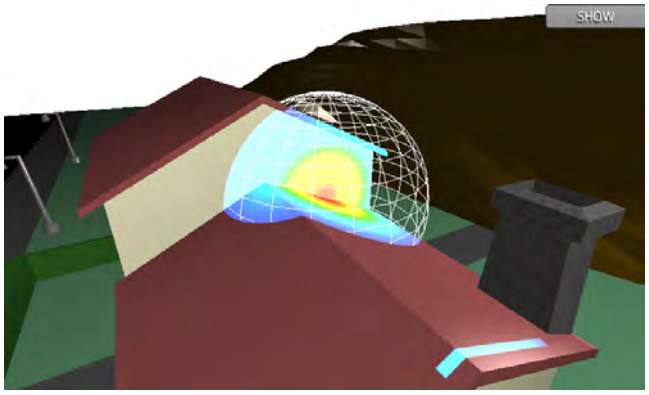

Figure 9: *If the artist of an object has reused areas of the original texture for different areas of the object, using the same coordinate mapping may result in artifacts with our approach (see lower right corner).*

6.5 Reducing Number of Viewing Events

For real-time visualization, one option is to visualize the incoming attention data as it comes, effectively showing the dwell time on the target objects and not fixation data in the strict sense. As eye-tracking data comes in with 60 Hz with our system, we have only about 17 ms to update the heat map for the next frame. The second option would be to wait for the more high-level fixation events and only then update the visualization. As fixations start with duration of about 90 ms and can go up to several hundred ms, the performance, but also the latency of the visualization, would increase five times or more. The presented shaders support both approaches by setting the viewing_duration of the viewing events accordingly.

6.6 Performance Restrictions based on Memory-transfer between GPU and CPU

During normalization our approach needs to access the memory buffers on the GPU to extract the current maximum attention value. For this we use the function glReadPixels. However, to access the buffer, CPU and GPU have to synchronize, which leads to a wait on side of the CPU. To overcome this, we could also use Pixel Buffer Objects (PBOs) to read the data asynchronously. Then, however, by the asynchronous reading it could happen that the new global maximum is not available for the rendering of the directly adjacent frame. This results in a visualization that is actually representing a slightly higher attention than what is represented in the Attention Textures. Note that this is only an artifact of the real-time visualization using the shader. The represented values in the Attention Textures are always correct.

7 Conclusion

The presented approach for generating realistic 3D heat maps for visualizing visual attention extends the state-of-the-art in several ways. It can handle 3D scenes and, based on 3D point of regards, provides more realistic results by respecting full and partial occlusions. Our 3D model for attention spreading supports binocular perspectives and depth of focus.

The target objects can also be pre-selected, as attention is stored on a per-object level. This allows for covering large exhibitions in which not the architectural details, but certain historic artifacts are of interest. As the attention is stored object-centered in the texture of the object, the presented approach can handle moving or even deforming objects without any problem. It is also, so far as we know, the first approach that can handle this taking into account previous object transformations correctly. If, however, new vertices are added to the object during transformation, it has to be ensured that the texture mapping is preserved and new areas are added without creating texture overlaps.

There are several ways to handle multiple viewers. A direct approach is to include all viewing events over all viewers in the computation process. Alternatively, a set of Attention Textures can be created for each viewer and Attention Textures for a specific object can be aggregated by simply adding up the per-pixel values. This way it is also easily possible to create visualizations for different subgroups (based on gender, age, etc.).

One feature of using textures for representing the distribution of attention an object has received is that it allows for an efficient analysis of aggregations over multiple objects. If one is only interested in typical distributions of attention on a certain class of objects, for example a series of signs or information tables with multiple instances within a (digital) signage project, all instances could share the same attention texture and data will be aggregated efficiently.

Based on a multi-texturing approach, the results can easily be made persistent and used for high-quality renderings, as has been demonstrated (see Figure 6). For the presentation of the visualization, our approach no longer requires a special-purpose tool. The scenes with augmented heat maps can, e.g., easily be uploaded and visualized using WebGL on a standard website with current client devices.

For an evaluation of the performance of the system, please be refered to Pfeiffer and Memili [2015].

7.1 Future Work

Future research could evaluate the 3D model of attention spreading and improve upon this, e.g., by taking the panum area into account. In addition to that, the presented approach could be improved to support transparencies and reflections.

Our next steps will be to merge the presented approach with our EyeSee3D technology for eye tracking in 3D environments [Pfeiffer et al. 2016].

Additional material for this paper is available online on the accompanying website [Pfeiffer and Memili 2016].

Acknowledgements

This research was supported by the Cluster of Excellence Cognitive Interaction Technology 'CITEC' (EXC 277) at Bielefeld University, which is funded by the German Research Foundation (DFG).

References

BLASCHECK, T., KURZHALS, K., RASCHKE, M., BURCH, M., WEISKOPF, D., AND ERTL, T. 2014. *State-of-the-Art of Visualization for Eye Tracking Data*. The Eurographics Association.

BUCK, I., AND PURCELL, T. 2004. GPU gems, chapter ch. 37: A toolkit for computations on GPUs. *Addison-Wesley 2*, 626.

DIMITROV, R. 2007. Cascaded shadow maps. *Developer Documentation, NVIDIA Corp*.

DUCHOWSKI, A. T., PRICE, M. M., MEYER, M., AND ORERO, P. 2012. Aggregate gaze visualization with real-time heatmaps. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, ACM, New York, NY, USA, ETRA '12, 13–20.

EVERITT, C., REGE, A., AND CEBENOYAN, C. 2001. Hardware shadow mapping. *White paper, nVIDIA 2*.

HAMMER, J. H., MAURUS, M., AND BEYERER, J. 2013. Real-time 3D gaze analysis in mobile applications. In *Proceedings of the 2013 Conference on Eye Tracking South Africa*, ACM, New York, NY, USA, ETSA '13, 75–78.

LAURITZEN, A., SALVI, M., AND LEFOHN, A. 2011. Sample distribution shadow maps. In *Symposium on Interactive 3D Graphics and Games*, ACM, New York, NY, USA, I3D '11, 97–102.

MAURUS, M., HAMMER, J. H., AND BEYERER, J. 2014. Realistic heatmap visualization for interactive analysis of 3D gaze data. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, ACM, New York, NY, USA, ETRA '14, 295–298.

PFEIFFER, T., AND MEMILI, C. 2015. GPU-accelerated attention map generation for dynamic 3D scenes. In *Proceedings of the IEEE VR 2015*, IEEE, T. Hllerer, V. Interrante, A. Lcuyer, and J. E. S. II, Eds., IEEE, 257–258.

PFEIFFER, T., AND MEMILI, C., 2016. Companion website to this paper: http://etra2016heatmap.eyemovementresearch.com/.

PFEIFFER, T., RENNER, P., AND PFEIFFER-LESSMANN, N. 2016. Model-based real-time analysis of mobile eye-tracking on static and dynamic three-dimensional scenes. In *ETRA '16: 2016 Symposium on Eye Tracking Research and Applications Proceedings*, ACM Press. DOI: http://dx.doi.org/10.1145/2857491.2857532.

PFEIFFER, T. 2012. Measuring and visualizing attention in space with 3D Attention Volumes. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, ACM, New York, NY, USA, ETRA '12, 29–36.

POMPLUN, M., RITTER, H., AND VELICHKOVSKY, B. 1996. Disambiguating complex visual information: Towards communication of personal views of a scene. *PERCEPTION-LONDON- 25*, 931–948.

STELLMACH, S., NACKE, L., AND DACHSELT, R. 2010. 3D attentional maps: Aggregated gaze visualizations in three-dimensional virtual environments. In *Proceedings of the International Conference on Advanced Visual Interfaces*, ACM, New York, NY, USA, AVI '10, 345–348.

WOLFF, D. 2011. *OpenGL 4.0 shading language cookbook*. Packt Publishing Ltd.

ZHANG, F., SUN, H., XU, L., AND LUN, L. K. 2006. Parallel-split shadow maps for large-scale virtual environments. In *Proceedings of the 2006 ACM International Conference on Virtual Reality Continuum and Its Applications*, ACM, New York, NY, USA, VRCIA '06, 311–318.