Real-Time Tracking of Visually Attended Objects in Virtual Environments and Its Application to LOD

Sungkil Lee, Student Member, IEEE, Gerard Juonghun Kim, Member, IEEE, and Seungmoon Choi, Member, IEEE

Abstract—This paper presents a real-time framework for computationally tracking objects visually attended by the user while navigating in interactive virtual environments (VEs). In addition to the conventional bottom-up (stimulus-driven) saliency map, the proposed framework uses top-down (goal-directed) contexts inferred from the user’s spatial and temporal behaviors and identifies the most plausibly attended objects among candidates in the object saliency map. The computational framework was implemented using GPU, exhibiting high computational performance adequate for interactive VEs. A user experiment was also conducted to evaluate the prediction accuracy of the tracking framework by comparing objects regarded as visually attended by the framework to actual human gaze collected with an eye tracker. The results indicated that the accuracy was in the level well supported by the theory of human cognition for visually identifying single and multiple attentive targets, especially owing to the addition of top-down contextual information. Finally, we demonstrate how the visual attention tracking framework can be applied to managing the level of details in VEs, without any hardware for head or eye tracking.

Index Terms—Visual attention, saliency map, bottom-up feature, top-down context, virtual environment, level of detail.

1 INTRODUCTION

Knowing the region or objects to which a user attends in an image yields a number of advantages for creating an effective interactive virtual environment (VE). For example, once a region or objects that are visually focused on are identified, the rendering fidelity can be selectively controlled for lower computational costs or better perceptual effects. This simple idea resulted in a number of useful computer graphics and virtual reality (VR) techniques, including level-of-detail (LOD) management [1], [2], mesh simplification [3], information culling in distributed VEs [4], peripheral degradation (or foveation) [5], and global illumination [6], [7].

A straightforward way to locate a visually attended region or objects in an image is to rely on an eye tracking device. However, such devices are still expensive and uncomfortable to wear and require a cumbersome calibration procedure. Moreover, most tracking devices do not allow the wearer to move and thus are not appropriate for dynamic VR applications. A prominent alternative is to apply principles learned from human visual perception to computationally estimate a region or objects, which the user might be looking at and focusing upon.

It is well known that human visual attention is affected by both automatic capture of bottom-up (stimulus-driven) salient stimuli and volitional shifts guided by top-down (goal-directed) factors [8], [9]. Color and intensity in images are common examples of bottom-up stimuli [10]. Prior knowledge, memories, and goals can be top-down factors [11], [12]. Among the numerous bottom-up attention models that have been conceived in the visual perception literature, the feature integration theory of Treisman and her colleagues [10] is most widely recognized and accepted. In this theory, an area in an image is called visually salient when it stands out relative to its surrounding neighborhood. A 2D map that represents salient regions or objects in the image is called the saliency map. In addition, it has been verified that the saliency of a region is determined by integrating the low-level features such as color and intensity.

Inspired by the feature integration theory, several computational models have been proposed to estimate visual saliency in an image [13], [14], [15], [16], [17]. In particular, the model of Itti et al. [16] is relatively simple and has been used frequently in practice. It computes individual contrast maps for bottom-up features using the center-surround difference and then integrates them into a final saliency map using the Winner-Take-All network [13]. Given an image, a center-surround difference refers to a difference between coarser and finer images, both generated from the original image using image pyramids [16] or the difference-of-Gaussian filter [18]. This framework has been applied and extended to various applications, including...
object recognition [19], video compression [20], and video summarization [21].

Once salient regions are determined based on the preattentive bottom-up features, top-down factors may further guide a user to select one region to focus on. Unlike the bottom-up features, this top-down guidance is task dependent [8], [12], [22], [23]. For example, if red and green apples are in sight under no other specific conditions, a user is likely to look at either one (since red and green are opponent channels in the visual cortex [24], they have comparable bottom-up saliency levels). However, a particular task (e.g., look for a red object) given to the user would direct the user’s attention to the red apple. Such task dependence has been a limiting factor for incorporating the effect of top-down factors into visual attention estimation. Cater et al. introduced the importance factor of objects relative to a task in computing saliency [1], but the factor values had to be set manually. Features such as object distance from a viewer, image coverage [25], and novelty [6], [7] have also been treated as top-down factors. However, since these factors generally do not reflect the user’s intention, they are more appropriate to be classified as bottom-up features. Recently, Peters et al. [26] demonstrated the effect of task-dependent influences in video-games using a learning methodology based on the bottom-up features. Since their consideration of top-down influence is related to the specific task or environment, prediction of the individual intention of a user does not seem straightforward. Compared to the bottom-up features, few attempts have been made to systematically take into account the effect of the user’s intention for predicting attention.

Confirming whether a computational model for visual attention agrees to actual human attention is crucial to the usability of the model in applications. Ouerhani et al. [27] compared the saliency model of Itti et al. [16] to actual human attention using an eye tracker and confirmed the existence of correlated regions in images. Santella et al. applied the bottom-up saliency model to nonphotorealistic scenes and compared its prediction performance with that of an eye tracker [28]. However, compared to the abundant theories and applications of the saliency map, efforts toward their formal validation have been somewhat insufficient.

In this paper, we present a real-time computational framework for predictive tracking of visual attention in dynamic and interactive VEs. Despite many potential applications such as human gaze estimation, perceptually based rendering, automatic foveation, and a depth-of-field effect, computational saliency models adequate for dynamic VEs have received relatively little attention. In what follows, we enumerate the primary obstacles that have impeded the appearance of such models and our remedies for each.

First, applications in dynamic VEs require temporally coherent saliency prediction of objects, rather than pixels. Previous cognitive studies revealed that visual attention is object based rather than location based and that it changes in association with the movements of objects [29], [30], [31]. This inspired the idea of attended object tracking. However, most saliency computation models only consider pixel-based regions and cannot guarantee the coherence of object saliency values during interactive simulation. Our computational framework provides the estimation of visual attention levels of objects in a VE and also allows their smooth tracking over time.

Second, the region of interest derived from a saliency map based on the conventional bottom-up features does not reflect a user’s volitional factors. This may result in incorrect attention prediction, since the top-down factors play a vital role in visual attention as reported in the previous literature [8], [11], [12]. Our attention tracking framework employs top-down contexts for the user’s spatial and temporal intention inferred from the user’s behavior during navigation in a dynamic VE.

Third, the computational cost for generating a saliency map has been relatively expensive. To resolve this issue, Longhurst et al. [7] suggested using the Graphics Processing Unit (GPU) to accelerate computation. Our work, which has been performed independently from that of Longhurst et al., has followed a similar but improved approach by using the hardware mipmap generation of the GPU. The implemented framework shows a remarkable real-time performance that is fast enough to be used for interactive VEs.

Finally, the accuracy of visual attention estimation needs to be validated for such computational methods to be used in real applications. We conducted a user experiment using an eye tracker to evaluate the accuracy of our attention prediction and to assess the relative contributions of the bottom-up and top-down factors.

In addition, we demonstrate how to use the attention estimation framework for improving perceptually based rendering methods, with the management of level of detail (LOD) as an example. From an early multiresolution representation for a single object (later called the discrete LOD or DLOD) [32], techniques for LOD management have evolved into numerous continuous and view-dependent methods (e.g., see [33], [34], and [35]). The most important aspect in LOD management is the decision logic for switching between different model resolutions during rendering. In principle, the more faithful LOD selection criteria to the user’s attention, the less degradation in the perceptual quality will appear. For example, even though using simple metrics such as distance, size, or priority result in reasonable performance improvement [36], the perceptual degradation is easily noticeable since they are weakly related to the human visual attention. More elaborate metrics have been tried to maximize the performance gain based on an assumption that the user’s gaze is fixed at the screen center [37] or by using relatively simple information such as user-defined importance and a bottom-up saliency model [6]. Nonetheless, they do not take the user’s volitional information into account, often making the use of low-resolution models perceptible. Other perceptually accurate approaches, e.g., that based on the peripheral vision, assume that a focused object/area by the user is known a priori [36]. Except for the simplest ones, all of these measures require an accurate identification of an attended object/area. However, this has not been possible without an eye-tracking device prior to our attention tracking framework.
By simply using the attention values of objects as their fidelity levels for LOD, our attention tracking framework can be easily integrated into the existing LOD switching techniques. In this article, we associate the attention tracking framework with the “Unpopping” LOD [38] that is a recent image-space blending technique for coping with the “popping” effect during the discrete LOD switching.

The rest of this paper is organized as follows: Section 2 introduces the overall structure of our visual attention tracking in interactive VEs. Then, Section 3 explains how a bottom-up saliency map is constructed using the GPU in real time. Section 4 presents the object-level attention tracking modulated by top-down contexts. Section 5 describes the implementation details and reports the computational performance. In Section 6, the accuracy of attention prediction is experimentally validated. Section 7 demonstrates LOD management as an example of practical applications. Finally, Section 8 concludes this article, including a plan for future work.

2 OVERVIEW OF FRAMEWORK

The overall procedural flow of our framework for visual attention tracking is summarized in Fig. 1. The framework consists of two components, one for building a bottom-up saliency map (upper block in the figure) and the other for modulating the saliency map using top-down contexts (lower block in the figure). In essence, our bottom-up saliency map extends that of Itti et al. [16] using two image features (luminance and hue) and three 3D dynamic features (depth, object size, and object motion). With 3D geometric models, simulation models, and an RGB image rendered from the models, feature maps for luminance, hue, depth, size, and motion are generated as image pyramids (Step 1 in Fig. 1). Then, the image pyramids are converted by the center-surround difference to build contrast maps (alternatively called the conspicuity maps in [16]) for each feature (Step 2 in Fig. 1), which indicate regions with abrupt changes of pixel values in the corresponding feature map. This procedure is computed in real time using a GPU program. Finally, the contrast maps are linearly combined into a single map that represents the saliency of each pixel obtained from the bottom-up features (Step 3 in Fig. 1).

The first step for top-down contextual modulation is to convert the pixel saliency map to an object saliency map, such that pixels corresponding to an object have the same saliency value (Step 4 in Fig. 1). The relevance of an object to a given task is also considered in this step. Then, top-down contextual (spatial and temporal) factors are computed, and the object saliency map is modulated with them (Step 5 in Fig. 1). The spatial context refers to the importance of the spatial layout of objects for predicting the observer’s attention, which is more effective for short-term tasks. The temporal context reflects the evolution of the spatial context associated with long-term goals. Finally, the map is linearly filtered for each object using the Kalman filter for smooth tracking (Step 6 in Fig. 1). In subsequent sections, detailed explanations for each step are provided.

3 REAL-TIME BOTTOM-UP SALIENCY MAP

This section presents the details of our computational framework for building a pixel-level saliency map in real time. Inputs into the framework are rendered color/depth images, 3D object models, and a simulation model, all of which are commonly required and available in a VE.

3.1 Bottom-Up Features and Their Image Pyramids

An initial step for building a saliency map is to construct feature maps using the five low-level features: luminance, hue, depth, size, and motion. Previous neurological studies showed that all these features are preattentive (see [10] for luminance, [39] for hue, [40] for depth, [10] for size, and [41] for motion). Compared to the original image features used by Itti et al. [16], we do not use the edge orientation feature since our approach focuses more on objects than image-level details. Instead, more relevant features for predicting object-level saliency in VEs such as depth, size, and motion are adopted. These feature values can be specified either for each pixel (luminance, hue, and depth) or for each object (size and motion).

The first two feature maps for luminance \( B_l \) and hue \( B_h \) are taken from the luminance and hue components in the Hue-Luminance-Saturation (HLS) color model converted from the original RGB image. Each pixel value for the depth map \( B_d \) is obtained from the z-buffer and normalized as

\[
B_d = \frac{z - z_{\text{near}}}{z_{\text{far}} - z_{\text{near}}},
\]

where \( z, z_{\text{near}}, \) and \( z_{\text{far}} \) are the 3D, near, and far clipping depths, respectively.

A feature map for size \( B_s \) is defined at the object level by considering the image coverage of an object. For object \( k \), pixel values of the object are set to

\[
B_s(k) = \frac{\text{number of pixels belonging to object } k}{\text{total number of pixels in the image}}.
\]

This method is effective when the object size is larger than the view volume, or an object is partially culled by the
viewport. The number of pixels corresponding to each object is counted using the item buffer [42].

A motion feature map \( B_m \) represents the velocity of an object, obtained from the difference of 3D positions at consecutive simulation frames \( \tau \) and \( \tau - 1 \). It is computed for object \( k \) as

\[
B_m(k) = \| \mathbf{p}^\tau(k) - \mathbf{p}^{\tau-1}(k) \|,
\]

where \( \mathbf{p} \) denotes the position of a vertex at the outermost position from the object’s center. This allows us to consider both the translation and the self-rotation of the object.

Each feature map is successively downsampled and converted into a set of lower resolution images, forming image pyramids from the finest original map (level 0) to the coarsest (level 6). These image pyramids are used for the center-surround difference operation to build contrast maps at the next step. Note that the processes are executed in a single batch using the hardware mipmap generation capability in the graphics hardware for real-time computation (see Section 5 for more details).

### 3.2 Pixel-Level Saliency Map

The five feature maps are converted to local contrast (or conspicuity) maps, \( C_l, C_h, C_d, C_s, \) and \( C_m \), via the center-surround difference [16]. This is an operation that detects locations standing out from their surroundings. The center-surround difference for a contrast map, \( C_f \) \( \langle f \in \{l, h, d, s, m\} \rangle \), is calculated as

\[
C_f = \frac{1}{6} \sum_{i=0}^{2} \sum_{j=0}^{2} \left| B_f^i - B_f^{i+j} \right|,
\]

where \( B_f^i \) and \( B_f^{i+j} \) represent feature maps at pyramid level \( i \) (finer) and \( i + j \) (coarser), respectively. This operation is quite effective at finding contrasts and is widely used for bottom-up saliency map computation.

The contrast maps are merged into a single topographical saliency map, \( S_p \), by linearly combining them as

\[
\hat{S}_p = \sum_{f \in \{l, h, d, s, m\}} w_f C_f,
\]

where \( w_f \)'s are linear combination weights. Even though various schemes can be used for determining the weights [16], most of them are not suitable for real-time use. In our framework, the weights are set to

\[
w_f = \frac{1}{\max_{u,v} C_f(u,v)},
\]

where \( (u,v) \) represents a pixel position in \( C_f \). The weights are for balancing the differences in the dynamic ranges of the features. Finally, \( S_p \) is normalized to \( \hat{S}_p \) (the pixel-level saliency map), such that each pixel value in \( S_p \) ranges in [0, 1].

### 4 Modulation by Top-Down Contexts

Top-down contextual information provides additional criteria for selecting objects among visually salient candidates found from the bottom-up features. With navigation in a VE as a primary task, we have modeled a few high-level contexts (i.e., task-related object importance and spatial and temporal contexts) and included them in our computational framework for improved and plausible attention estimation.

#### 4.1 Object-Level Attention Map

We first convert the pixel-level saliency map \( S_p \) to an object-level saliency map \( S_o \). As noted in [29], [30], and [31], an object-level saliency map is more appropriate for tracking and applications in VEs. This is achieved by averaging the pixel values of each object as

\[
\hat{S}_o(k) = T_i(k) \frac{1}{n(k)} \sum_{(u,v) \in \text{object } k} S_p(u,v),
\]

where \( n(k) \) is the number of pixels corresponding to object \( k \), \( (u,v) \) is a pixel position, and \( T_i(k) \) is the user-defined task-related importance of object \( k \). The pixels that are associated with object \( k \) are determined using the item buffer [42]. The task-related object importance is used for excluding unimportant background objects (e.g., wall, floor, sky, and seawater) from consideration.

The object saliency map is further elaborated with spatial and temporal contexts \( (T_s, T_t) \) that are used to infer the user’s intention during interactive navigation. The models of the spatial and temporal contexts are described in the subsequent sections. Once the spatial and temporal contexts are determined, the final object attention map, \( S_o(k) \), is computed as

\[
S_o(k) = (T_s(k) + T_t(k)) \hat{S}_o(k).
\]

Note that \( S_o(k) \) values may become temporally unstable due to the bilinear magnification in the texture lookup used in the center-surround difference operation, which is a common problem with texture magnification. The blocky artifacts due to the magnification can result in abrupt changes of saliency values. In order to smooth the changes, the attention levels of objects are postprocessed using a linear Kalman filter [43].

#### 4.2 Spatial Context Based on User Motion

Typical models of spatial perception such as the topological three-stage model (landmark, route, and survey knowledge) require a hierarchical cognitive map of objects [44], [45]. These representations are related to long-term goals rather than immediate responses of perception-action. Such models are usually too complex to be used in practice for predicting a user’s intention.

Assuming that landmarks (foreground objects) exist in a VE, one effective way to estimate the general area of a user’s interest, without any cognitive map, is to find the moving direction of the user’s egocentric view based on atomic interaction data. Hinted at previous findings of Cutting and Vishton [46], our method is to model three spatial behaviors of an observer who navigates in a VE.

Let \( x \) be a distance from the observer to an object in the 3D scene, \( y \) be a normalized distance between the center of a screen and the object in the screen coordinates, \( v \) be the viewing direction of the observer, and \( w \) be the moving direction of the observer (see Fig. 2 for conceptual illustrations). \( \Delta x = x^\tau - x^{\tau-1} \) is the difference in \( x \) between two consecutive simulation frames, where \( \tau - 1 \) and \( \tau \) are
the corresponding time indices. First, we note that observers tend to situate themselves so that they can see objects in the center of a screen during navigation. It follows that objects far from the screen center are not likely to be attended to. A modulation factor for this behavioral pattern is expressed as an exponential decay of the distance between the screen and object centers: \( e^{-c_yx/c_y^2} \), where \( c_y \) is a scaling constant. Second, observers prefer to maintain a proper distance from objects of interest, which requires modulating the spatial emphasis of objects based on the distance from an observer. Our modulation factor for this pattern is in accordance with the Weibull distribution: 
\[
(x/c_x)e^{-(x/c_x)^2},
\]
where \( c_x = D/0.707 \) is a scaling constant that is determined from a desired distance (\( D \)) of maximum spatial emphasis. Third, when observers move forward (\( \mathbf{v} \cdot \mathbf{w} > 0 \); also see Fig. 2), they usually approach objects that they want to see. Therefore, objects that are becoming more distant from the observer (\( \Delta x > 0 \)) are very unlikely to receive any attention from the observer. Combining the three modulation factors, the spatial context model for object \( k \) is defined as

\[
T_s(k) = \begin{cases} 
0, & \text{if } \mathbf{v} \cdot \mathbf{w} > 0 \text{ and } \Delta x > 0, \\
\frac{x}{c_x}e^{-(x/c_x)^2}, & \text{otherwise},
\end{cases}
\]

where \( c_x \) represents a scaling constant to maintain \( T_s(k) \) within \([0, 0.5]\).

4.3 Temporal Context

High spatial context values observed for an object during a certain period of time imply that the object has been followed by the user. Consideration of this temporal property is useful for estimating the long-term intention of the user, whereas an immediate task is mediated by the spatial context. To reflect this property, we define temporal context, \( T_t(k) \), for object \( k \) using the running average of the spatial contexts as

\[
T_t(k) = \frac{1}{\lambda} \sum_{\tau=\tau_0-\lambda+1}^{\tau_0} T_s^\tau(k),
\]

where \( \lambda \) denotes duration for controlling the long-term interest, and \( \tau_0 \) and \( T_s^\tau \) are the current simulation frame and the spatial context at \( \tau \), respectively. An additional role of the temporal context is to compensate for the erroneously derived spatial context values (e.g., unintended control errors).

5 IMPLEMENTATION DETAILS AND COMPUTATIONAL PERFORMANCE

The proposed framework for real-time visual attention tracking was implemented using the OpenGL Shading Language (GLSL) and the OpenSceneGraph on a 3.2-GHz Pentium 4 PC with a GeForce 7900GTX. In the preprocessing stage, objects represented in 3D models were segmented in a scene graph, and a unique ID and task-related importance \( (T_t) \) were assigned to each.

During runtime, the computational procedure shown in Fig. 1 is applied at every simulation frame. The computational steps consist of four stages:

1. updating the object-level features and spatial and temporal contexts,
2. building a pixel saliency map using the GPU acceleration,
3. converting the pixel saliency map into an object saliency map and modulating it with the top-down contexts, and
4. storing the result in an object attention list and postprocessing it using a linear Kalman Filter.

In the first stage, the simulation engine updates the object-level features (the motion and size) and the spatial and temporal contexts based on the data recorded in the previous simulation frames. The spatial and temporal contexts for each object are computed using the object position in the world coordinate system and the viewing matrix from user interaction.

In the second stage, the graphics engine renders bottom-up feature maps and contrast maps using the GPU to construct a unified pixel-level saliency map. First, a typical RGB image is rendered from the 3D object models. Since three channels (usually for red, green, and blue) in a single texture can be simultaneously processed in a fragment shader, we render two feature map textures for the five bottom-up features, LH \((B_l, B_h, \cdot)\) and DSM \((B_d, B_s, B_m)\). To obtain contrast maps for the five features, we build two feature mipmap maps using the hardware mipmap generation capability and compute the center-surround difference using the mipmap. A sample fragment program for the
center-surround differences is shown in Fig. 3. The texture lookup operation for a coarser scale corresponds to the magnification of an image, and thus, it significantly improves computational performance, achieving real-time construction of the pixel-level saliency map. Then, the pixel saliency map \( S_p \) is built by linearly combining the two center-surrounded textures with the normalization weights. The weight of a contrast map is the reciprocal of a maximum pixel value in the map.

The third stage is to convert the pixel-level saliency map into an object-level saliency map. For this, the correspondence between pixel position \((u,v)\) and object \(k\) is found using the item buffer [42], which is an image, where each pixel contains the ID of the object that the pixel belongs to. Using the item buffer and the top-down contexts \((T_i, T_s, \text{and } T_t)\), the attention value of each object can be computed. For efficiency, the object-level attention map is stored in a linked list.

Finally, the computed attention value of each object is smoothed and tracked using a discrete Kalman Filter [43], based on the position and velocity state model.

Fig. 4 shows examples of intermediate and final output images of our attention tracking framework applied to static and dynamic environments. Object-level lists \((S_o, T_s, \text{and } T_t)\) were represented in maps for illustration. In these examples, a number of bottom-up candidates for attentive objects are initially suggested, with similar saliency levels in the contrast maps. Thus, the pixel-level saliency maps \( S_p \) alone do not clearly single out the most salient objects. With the aid of the top-down contexts \((T_s \text{ and } T_t)\), the most attentive objects are ultimately selected in a manner which is in greater correspondence with our visual attention (to be validated in Section 6).

In order to compare the computational performance of our framework to a typical CPU-based method and the most recent GPU-based method [7], the same algorithms were implemented using the OpenCV toolkit and GLSL. To remove the effect of the polygonal model size and the number of objects, we excluded the time required for model rendering (i.e., RGB, depth, and item buffer images). Also, the computation time for the top-down contexts was not included, since the top-down contexts are unique in our framework. In practice, the time required to compute the top-down contexts is negligible compared to that of the bottom-up saliency map (e.g., less than 0.3 msec for the virtual undersea in Fig. 4). The comparisons were performed for four saliency map sizes \((64 \times 64, 128 \times 128, 256 \times 256, \text{and } 512 \times 512)\). Fig. 5 shows the results of comparison. The speedup compared to Longhurst et al.’s method is between 1.57 times (for the 64 \times 64 image) and 1.15 times (for the 512 \times 512 image). The speedup compared to the CPU-based method is between 2.20 times (for the 64 \times 64 image) and 6.27 times (for the 512 \times 512 image).
Thus, we project that real-time use of our framework is possible with up to 256 × 256 saliency maps. Even if the model rendering times excluded in the comparison were to be included, our framework could produce 256 × 256 saliency maps in real time for 3D environments consisting of up to a million polygons (attention computation ≈ 5.68 msec, a total of 30 frames per second).

Recently, Longhurst et al. [7] reported a similar saliency map computation method that generates the Gaussian pyramids with multipass rendering and performs the center-surround difference in a GPU program. Compared to that of Longhurst et al., our work is more comprehensive in the following aspects. First, our implementation using the hardware mipmap generation for generating image pyramids requires single-pass rendering and is more efficient than that of Longhurst et al. Second, we use more comprehensive features, including top-down contexts that were not considered in their study. Finally, our attention prediction is for objects, unlike the study of Longhurst et al. In pixel-level saliency maps, both saliency prediction techniques may suffer from instability, since the magnification by texture lookup is common to both. However, we greatly suppress such instability in the object-level attention map.

6 EXPERIMENT FOR ATTENTION PREDICTION ACCURACY

This section reports the design and results of a user experiment conducted to validate the accuracy of our attention prediction compared to the actual human gaze pattern.

6.1 Methods

A participant’s eye movements were recorded using a monocular eye tracking device with a 60-Hz sampling rate and 640 × 240 video resolution (Arrington Research, Inc.; see Fig. 6a). A 42-inch LCD display with a resolution of 1,600 × 900 was used for the presentation of visual scenes (see Fig. 6b). The participant, wearing the eye tracker, was seated at approximately 1.5 m in front of the display in a lighted room for both a wide field of view and precise eye tracking.

A dynamic and a static VEs were used in the experiment. The dynamic environment modeled a virtual undersea (shown in Fig. 7a), where 30 animated fishes (of six types, with five fishes per type) were swimming. The static environment was a virtual art gallery shown in Fig. 7b.

There were 16 paid subjects (15 male and 1 female) who participated in the experiment. Their ages varied from 18 to 36 years, with a mean of 25.1 years. All participants had normal or corrected-to-normal vision. They were assigned to two types of tasks, free navigation and visual search. For free navigation, the participants were asked to simply move around the VEs and look at virtual objects using the keyboard for controlling navigation. For visual search, objects in the two VEs contained numbered tags on their body (see Fig. 8 for examples), and the participants were instructed to find objects that had specific numbers. For objects in the undersea, the numbers were 5, 15, 25, and 35. For the gallery, they were 7, 17, 27, and 37. The participants were asked to press the space bar when they found objects with the specified numbers. This instruction was given solely to maintain the participants’ concentration on the search task. Whether they indeed found the objects was irrelevant for the purpose of the experiment. With the two environments and the two tasks, the experiment used a 2 × 2 within-subject design.

Prior to an experimental session, a participant was briefed about the experimental procedure and undertook a training session to learn the navigation scheme in the art gallery model where no objects appeared. A control mechanism with three degrees-of-freedom (forward/backward translation and turns for yaw and pitch) was used for navigation. Four (2 × 2) main sessions followed the training session, and their presentation order was balanced using Latin squares. Each session started with the calibration of the eye tracker to map screen-space points in the 5 × 5 grid to eye-space coordinates. In order to find eye-space coordinates, an oval-fitted center of a pupil or a glint were extracted from an
input video of the eye tracker. After the calibration, the participant was instructed not to move his/her head, and this was facilitated with a chin rest (see Fig. 6a). The task for each session was verbally explained to the participant at the beginning of the session. Each session lasted for three minutes, and the participant took a rest for a few minutes before starting another session.

6.2 Data Analysis

For each experimental condition and each participant, 10,800 (60 pt/sec × 180 sec) screen positions that were stared at by a participant were measured with the eye tracker and recorded in terms of normalized screen-space coordinates: (0.0, 0.0) to (1.0, 1.0). These data were classified into four categories (blink, saccadic, drift, and fixation) according to the aspect ratio of a fitted ellipse and eye movement velocity. If the aspect ratio of an ellipse fitted to the eye was less than a threshold (0.7 for our data analysis), the eye movement was considered to be a blink. If not, the eye movement velocity was used for further classification. The velocity value is simply the difference between the current and the last gaze points, i.e., the change in the normalized position of gaze. A frame that showed eye movement velocity slower than 0.03 was regarded as fixation, a frame with eye velocity faster than 0.1 as saccadic, and a frame with eye velocity in between as drift. Only fixation and drift were used for analysis.

Three quantitative measures, $A_1$, $A_2$, and $A_3$, were defined in order to assess the accuracy of attention prediction. In each simulation frame, if one of the objects that the user’s gaze lingered on was the object predicted to be attended to, the attention estimation was considered correct for that frame. If so, the total number of frames with correct attention estimation was counted up. Note that during the counting, frames that showed only background objects were excluded, since no attentive objects were available in the participant’s sight. The number of frames with correct prediction was divided by the total number of frames that contained foreground objects, and the result was represented as $A_1$. Accuracies where two and three of the most attentive objects were compared with the participant-stared object were also computed and denoted by $A_2$ and $A_3$. These two measures accounted for the near-misses of attention prediction that may occur due to the human attention simultaneously laid on multiple objects [30], [31], [47], [48] or the limited precision of the eye tracker.

---

**Fig. 9.** Means and standard errors of the three estimation accuracies ($A_1$, $A_2$, $A_3$) for the task (free navigation versus visual search) and the environment (static versus dynamic) factors.

<table>
<thead>
<tr>
<th></th>
<th>$F_{1,15}$</th>
<th>$p$</th>
<th>$F_{1,15}$</th>
<th>$p$</th>
<th>$F_{1,15}$</th>
<th>$p$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K$</td>
<td>34.75</td>
<td>&lt;.0001</td>
<td>30.64</td>
<td>&lt;.0001</td>
<td>22.80</td>
<td>.0002</td>
</tr>
<tr>
<td>$V$</td>
<td>1.31</td>
<td>.2700</td>
<td>3.36</td>
<td>.0866</td>
<td>7.40</td>
<td>.0158</td>
</tr>
<tr>
<td>$K \times V$</td>
<td>3.59</td>
<td>.0776</td>
<td>6.93*</td>
<td>.0188</td>
<td>22.17*</td>
<td>.0003</td>
</tr>
</tbody>
</table>

Note: *$p < .05$.

We also investigated the relative contributions of features to attention estimation. The features were classified into three groups: image features ($B : \{B_1, B_3\}$), extended 3D/object features ($E : \{B_3, B_5, B_6\}$), and top-down context ($T : \{T_2, T_4\}$). To investigate the role of each feature group, attention maps were generated with and without the feature group, resulting in a total of eight ($2 \times 2 \times 2$) attention maps. Their estimation accuracies were compared using statistical analyses. Note that the baseline attention map, which corresponds to a map generated with no feature groups, cannot be clearly defined, since such a map contains no information about attentive objects. However, ignoring the baseline case would yield an unbalanced missing cell design, which makes the interpretation of main factor effects unclear in ANOVA [49]. To avoid this problem, we simulated the baseline attention map by randomly choosing a pixel on the screen as the most attentive pixel.

6.3 Results

When the attention map generated with all components is used, the overall average of the measured accuracies increased from 0.553 for $A_1$ to 0.811 and 0.914 for $A_2$ and $A_3$, respectively. The mean accuracies for the two independent variables, task ($K$) and environment ($V$), are shown in Fig. 9. The visual search task yielded higher accuracies than the free navigation task, with an overall mean difference of 0.096, whereas the effect of the environment on the prediction accuracies was rather weaker than that of the task.

These effects of task and environment were analyzed via a two-way within-subject ANOVA for each accuracy measure, $A_1$, $A_2$, and $A_3$, and the results are summarized in Table 1, where all the relevant statistics are shown. The effect of the task on all three accuracies was statistically significant, with very small $P$ values. The effect of the environment was statistically significant for $A_3$, but not for $A_1$ and $A_2$. Even for $A_3$, the $P$ value (0.0158) was much larger than the corresponding $P$ value of the task (0.002). The interaction effects between the task and the environment were statistically significant for $A_2$ and $A_3$, but not $A_1$. Simple main effect tests conducted as posthoc analysis showed that, for free navigation, the mean accuracies in the static environment were significantly higher than those in the dynamic environment (mean differences = 0.061 and 0.055, $p = 0.0070$ and 0.0004 for $A_2$ and $A_3$, respectively), whereas the differences between them were insignificant under the visual search task condition ($p = 0.7957$ and 0.7687 for $A_2$ and $A_3$, respectively).
The relative contributions of the feature groups (B, E, and T) to attention prediction were examined by three-way ANOVA. Since the environment factor had little effects on the accuracies, the data were collapsed across the task factor. For the two types of tasks, free navigation and visual search, we report the means of A1, A2, and A3 for the main factors, B, E, and T, in Fig. 10. As expected, overall accuracies for visual search were higher than those for free navigation, but other trends were similar. The existence of all feature groups resulted in significant increases of accuracies in all conditions. In particular, T group exhibited the largest increases. This indicates that the top-down contexts played an instrumental role in correctly identifying attended objects.

The means and standard errors for all eight combinations of feature groups are provided in Table 2. The data collected for the four conditions of the task and environment factors were collapsed into a single data pool for analysis. In the table, B = 0 implies that the B group features were not used for attention map generation and B = 1 that the features were used. E and T can be interpreted in a similar manner. The accuracies, except for the baseline, varied from 0.432 (only B for A1) to 0.918 (B and T for A3). We can also confirm that the accuracies showed the largest differences between conditions with and without the top-down contexts. A 2 × 2 × 2 ANOVA performed with B, E, and T as independent variables showed that all main and interaction effects were statistically significant (p < 0.0001) for all accuracies.

<table>
<thead>
<tr>
<th></th>
<th>B</th>
<th>E</th>
<th>T</th>
<th>A1</th>
<th>A2</th>
<th>A3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1.560(0.06)</td>
<td>0.272(0.10)</td>
<td>0.539(0.012)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0.432(0.011)</td>
<td>0.726(0.011)</td>
<td>0.866(0.008)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.599(0.011)</td>
<td>0.746(0.011)</td>
<td>0.880(0.008)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0.446(0.011)</td>
<td>0.734(0.011)</td>
<td>0.874(0.008)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0.561(0.016)</td>
<td>0.815(0.013)</td>
<td>0.915(0.008)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0.547(0.015)</td>
<td>0.808(0.012)</td>
<td>0.918(0.008)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.554(0.015)</td>
<td>0.812(0.012)</td>
<td>0.914(0.007)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.553(0.015)</td>
<td>0.811(0.012)</td>
<td>0.914(0.007)</td>
</tr>
</tbody>
</table>

6.4 Discussion

As discovered in earlier psychological studies [30], [31], [47], [48], human visual attention can be directed at multiple objects (or split focus) at the same time. Therefore, it is inherently difficult to precisely define a single object that a person stares at. Moreover, since the eye tracking device has inevitable errors (as high as 3 inches for a 42-inch screen), it was sometimes impossible to exactly pinpoint the attended object when several objects were closely located and overlapped on the screen. Considering all these facts, our attention tracking framework showed very promising results; the best accuracy was as high as 0.625 for a single attentive candidate and 0.945 for three candidates. Our framework can be used to find a single candidate (e.g., for direct gaze estimation required in depth-of-field rendering) and multiple candidates (e.g., for gaze-contingent LOD management and automatic generation of a camera path in large VEs).

As expected, the specific task (free navigation versus visual search) imposed on the participants resulted in statistically significant differences in attention estimation accuracies. During free navigation, the participants were often more interested in perceiving the spatial layout of an environment, rather than observing the details of objects. This must have degraded the overall accuracies for the free navigation task. The posthoc analysis for the interactions between the two factors showed that this phenomenon makes attentional capture more difficult in the dynamic environment. For visual search, the participants more focused on the objects rather than the overall environmental layout. Furthermore, even when their attention was disrupted by the abrupt appearance of objects that were highly salient in terms of the bottom-up features, their interest tended to immediately return to the currently pursued objects, due to their long-term goals (i.e., visual search). Such behavior is adequately reflected in our temporal context model, enabling good performance for the visual search.

Among the three feature groups, the top-down factors (T) were shown to make the greatest contribution to improving the attention prediction accuracies. This gives rise to an important insight for the use of a saliency map. That is, even though attentive candidates can be suggested from a saliency map generated with only bottom-up features, correct prediction of the most attentive objects is greatly facilitated by considering the top-down contexts related to a user’s intention. Therefore, it is necessary to categorize the common user behaviors in a VE (e.g., navigation, manipulation, and selection) and develop appropriate high-level context models to accomplish more precise prediction.

There was one notable top-down behavior associated with the novelty of objects encountered during the analysis of the users’ gaze patterns. The first spatial context that emphasizes the objects in the center of the screen was shown to be quite effective. However, after finishing the task (e.g., reading the numbers on the objects), the participant’s attention shifted from the objects in the center of the screen to the boundaries of the screen to find a new object to stare at, often moving to unexplored areas in the
VEs. Including this behavior in our spatial context model may significantly improve the accuracy of our framework. Another feature that is not included in our computational framework is the head movement of a user. Tracking the movement of the user’s head may result in the same improvement in the tracking accuracy, but this would require an additional device such as a camera or 3D tracker. Note that in the present experiment, the user’s head had to be fixed on the chin rest due to the restriction of the eye tracking device.

In addition, our experiment showed that a few features may have a key role in predicting attention. If such key factors are selectively included in attention estimation without using all the features, a reasonably high performance can still be attained (as demonstrated in Table 2), reducing the computational cost. Also note that for some VEs, bottom-up features may still be required, because top-down information and object-level segmentation are not always possible (e.g., an image-based VE).

We also tested the effects of the saliency map size on estimation accuracy. In general, using a small saliency map is expected to result in difficulties in correctly selecting small salient objects. Using the data collected for the visual search task in the virtual undersea, we tested four saliency map sizes (from 512 × 512 to 64 × 64) and computed the accuracies. The results showed very marginal differences below 1 percent. This is due to the fact that the VE used for the experiment included relatively large objects.

7 Example Application: Attention-Guided LOD Management

LOD management is one of the applications where rendering performance can be improved with the use of our attention tracking framework. In this section, we describe how to apply our framework to LOD management using “Unpopping LOD” [38]. The combined rendering system shows greater computational performance with little degradation in the perceptual quality.

7.1 LOD Management Using Attention Tracking Framework

The DLOD techniques are still widely used for VR and graphics applications, because of their simple implementation and decoupled process for mesh simplification and runtime execution. However, they have a common perceptual problem, “popping” effect, which is induced from the discrete model transitions between adjacent levels. In this article, we adapt “Unpopping” LOD (ULOD) recently proposed by Giegl and Wimmer [38], where one model of an object is opaque rendered, and the other model is semitransparently rendered for level switching. The two rendered images are alpha-blended according to the transition time period, effectively removing the popping effect.

A procedure of using ULOD with our attention tracking framework is straightforward. In an offline preprocessing step, we simplify an original object model into a few multiresolution models. Then, for every rendering frame at runtime, the attention level of each object is estimated by the attention tracking framework. This attention value is used to determine the fidelity level for each object, and the object is rendered based on its fidelity level. In other words, we simply use the attention level of an object as the metric for LOD management instead of the conventional metrics.

In principle, the computational performance gain of LOD management is in exchange for the possible degradation of rendering quality. Thus, psychophysical experiments are required to objectively assess the benefits of LOD management. However, since the experiments are dependent upon many environmental and subjective factors, the exact quantification of such advantages requires strictly controlled experiments, and unfortunately, their results are often impossible to be extended to general cases.

Alternatively, several researchers have proposed to use computational heuristics that approximate the expected perceptual benefit by considering various user-defined factors such as image coverage, semantic importance, focus, motion, and simplification accuracy [37], [50]. On the other hand, the value of object attention computed in our tracking framework already reflects most factors used in the heuristics. For example, the image coverage and motion of an object are included as bottom-up features. The semantic importance of an object is also incorporated while converting the pixel saliency map to the object saliency map. As a consequence, the attention value of our framework is highly correlated to the benefit heuristics, implying that relating the attention level to the degree of model simplification for LOD would result in images of high perceptual quality.

7.2 Cost Model of Attention-Guided LOD Management

The cost of rendering without LOD management can be represented in a similar manner to the cost function proposed by Gobbetti and Bouvier [36], [50] as

\[
\text{cost}_{\text{NOLOD}} = T_a + t \cdot 1, \tag{11}
\]

where \(T_a\) is the time required for initialization, finalization, and setup of objects, \(t\) is the vector of the rendering times of original object models, and \(1\) is a 1-vector. The rendering cost including LOD management with attention tracking can be expressed as

\[
\text{cost}_{\text{LOD}} = T_n + T_a + t \cdot (a + r). \tag{12}
\]

Here, \(T_a\) is the time required for computing object attention values, which is independent of the model complexity (e.g., texture memory transfer, computation of size and motion features, and top-down contexts). \(a\) is the vector of ratios of object rendering times for attention tracking (required for the rendering of color image and depth/item buffer) to the original. Each entry in the vector \(r\) is a simplification degree that ranges from 1 (the original model) to 0 (a no-polygon model).

In order to obtain sufficient performance gain, the cost gain by the use of LOD must be much greater than the additional cost required for the attention estimation such that \(t \cdot (a + r) \ll t \cdot 1\). A problem here is that our saliency map computation requires twice the time (color image and depth/item buffer) for model rendering (that is, by \(a\)), although \(T_a\) is practically negligible for a complex scene (as shown in Fig. 5, 0.97 and 1.9 msec for a saliency map of 64 × 64 and 128 × 128, respectively). If we render color
image and depth/item buffer using the original model, the computation cost for attention maps exceeds that of rendering without LOD management, because all entries in a are close to 2.

Thus, given a scene, in order to reduce the rendering cost of the feature maps, we suggest two techniques. First, we remove the model rendering for the color image by reusing the scene (color) image rendered in the previous rendering frame, which makes all entries in a close to 1. Nevertheless, the item/depth buffer still requires rendering of models. Second, we reduce the number of triangles (or vertices) processed in the depth and item buffer rendering by passing a much coarser model instead of the original one. This allows a to be much smaller than 1 and r. For instance, if we render the coarsest model with 1/256 triangles of the original model, each entry in a becomes close to 1/256. In this way, the computing cost for the attention map is significantly reduced, and overall rendering performance can be improved.

An important thing to note regarding the use of coarse models in saliency map computation is whether the result using the simplified models coincides with that computed using the original models. In order to examine the differences, we empirically compared the attention maps for five simplified levels of models; the model at level 0 \((L_0)\) is the original model, and the models at higher level are the quarter of one-level lower model for each. Fig. 11 shows the examples of scenes and object attention maps rendered using the original and simplified models. The object attention maps of the scene with raw models and simplified models seem to be almost identical. In order to observe the differences in more detail, the attention levels of attentive—an object attention value is greater than zero—objects were measured during a 30-second fixed-path navigation. The results showed that the order of attentive objects were exactly the same for all levels. The differences of the object attention values among the five groups were examined via paired t-tests. The differences among all the pairs were statistically insignificant (all \(P\) values < 0.0001). The means of differences ranged from 0.001 to 0.017 over the normalized scale. This arises from the characteristics of our framework, where the saliency map operates at reduced resolutions, and the objects in the test VEs are much larger than the polygons modulated by LOD. Furthermore, our framework relies more strongly on top-down contexts than bottom-up saliency.

### 7.3 Associating ULOD with Object Attention Values

For LOD management, the level of an object model to be rendered needs to be determined among its multiresolution models. The model level can be associated with the attention value of the object in a straightforward manner. Let \(M(k)\), \(L(k)\), and \(r(k, L)\) be the highest level (the coarsest resolution) model for object \(k\), the desired level of the object model, and the simplification degree of triangles (or vertices) at the level \(L\), respectively. For simplicity, we omit the object identifier, \(k\), in subsequent derivations. For the LOD ranging from 0 to \(M\), \(L\) can be linearly mapped to the attention value, \(S_o (\in [0, 1])\), as

\[
L = M(1 - S_o).  
\]  

Note that \(L\) is continuous. In this relation, an object with the highest attention value (i.e., \(S_o = 1\)) is rendered using its finest model (\(L = 0\)), and vice versa. This \(L\) value is passed to the ULOD algorithm for blending images rendered using the models at the two adjacent discrete levels. While the original ULOD was proposed for blending the images during a transition period [38], we use this scheme to achieve smooth transition along continuous object attention values.

The multiresolution models of each object are prepared offline prior to rendering. For each discrete level, \(L\), the simplification degree (i.e., the ratio of triangles in a simplified model to the original), \(r(L)\), should be passed to a specific simplification metric. In principle, \(r(L)\) should be carefully designed so that perceptual degradation due to the simplification is less perceptible to the human and that sufficient performance gain is achieved. Aggressive simplification may result in some degree of popping, in spite of the smooth transitions using ULOD. For a reasonable trade-off, we have experimentally chosen the following function:

\[
r(L) = \beta^{-L},  
\]  

where \(\beta\) is a base that determines the simplification degree and should be carefully chosen such that the transition between adjacent discrete levels does not cause a popping effect. \(\beta = 4\) was used in our experiment. A simplified model corresponding to \(r(L)\) is generated using Quadric error metric [51], which has been widely used for mesh simplification. Fig. 12 shows an example of simplified models at five levels for the Stanford Bunny models.
7.4 Performance Evaluation

We implemented and tested the proposed LOD management framework on the identical platform used in the accuracy validation experiment of Section 6. A 1024 × 768 display resolution and a 64 × 64 saliency map were used in the test. With a simple background model (floor and wall), the test scene consisted of 256 objects, which are the maximum capacity allowed in the 8-bit item buffer. Two kinds of scenes were used: one with the Bunny models and the other with the Dragon models. For each scene, the models were randomly varied in terms of luminance, hue, size, position, and rotation. Each model was simplified to five fidelity levels. From level 0 to 4, the numbers of triangles were 69,451, 17,364, 4,341, 1,086, and 272 for the Bunny model and 435,708, 108,928, 27,232, 6,808, and 1,702 for the Dragon model, respectively. Fig. 13 shows examples of the Dragon scene used in the experiment.

We measured the rendering cost (time) during the free navigation of the scene. Since all the objects did not appear simultaneously due to the culling, the true rendering cost depended on the number of objects visible at a specific view. During the navigation, we measured the rendering times for each frame and stored them in a separate list based on the number of visible objects. A large number of rendering time data (in total at least 3,000 samples) were collected and analyzed.

In performance analysis, the independent variable was the type of LOD: rendering without LOD (NOLOD) and rendering with four sets of LOD levels (ULOLO2, ULOL3, ULOL4, and ULOL5). For instance, ULOL3 represented ULOL applied to the three levels (e.g., 69,451, 17,364, and 4,341 triangle models for the Bunny scene). The other level sets were defined in a similar manner. For each condition, the coarsest one in the corresponding model set was passed to the attention tracking module.

The means and standard errors for each test condition and test scene are summarized in Fig. 14. Overall, as expected, the rendering costs of NOLOD were higher than those of the ULOLs, except for ULOL2. Compared to the Bunny scene, the rendering costs for the Dragon scene were more reduced. For both scenes, ULOL4 and ULOL5 showed high-performance improvements.

Fig. 15 shows the evolution of the rendering costs with respect to the number of rendered objects. For the Bunny scene, when the number of objects was roughly less than 20, NOLOD was better than those under all the ULOL conditions. As the number of objects gradually increased, all ULOL methods outperformed NOLOD. However, the performance gain of ULOL2 required significantly many objects (roughly more than 40 objects). For the Dragon scene, ULOL3, ULOL4, and ULOL5 exceeded NOLOD, even for scenes with a small number of objects, whereas ULOL2 showed only marginal improvement.

7.5 Discussion

The attention-guided LOD management system demonstrated the utility of our attention tracking framework. This attention-based metric can be easily combined with other conventional LOD switching metrics such as distance from the viewer, size in the screen, and eccentricity in the human fovea. Furthermore, other continuous and view-dependent LOD approaches can be used with our framework. In this sense, our framework can serve as a general solution to perception-based LOD management for VR applications.

As shown in Fig. 15, the computational performance of our LOD framework can be degraded for a scene with a relatively small number of objects, which is common to any LOD schemes. What matters is how to minimize the fixed overhead for LOD management. In our case, the overhead corresponds to the cost required for attention tracking, and the model rendering time takes the most of it. However, we anticipate that this limitation can be partially surmounted by utilizing only top-down contextual information without bottom-up saliency. As already shown in Fig. 10, the accuracy of our attention estimation framework depends more on top-down contexts than on bottom-up saliency. Top-down information can be obtained with virtually negligible computation time based on the configuration of objects in a VE. As a result, in low-end systems where a complete version of our framework cannot be used, the simplified attention tracking framework using only top-down contexts can be an attractive alternative.

We also performed preliminary subjective evaluation on perceived visual difference among the five conditions.
Most subjects reported that the model degradation was perceived slightly under ULOD4 and significantly under ULOD5 and not under any other conditions. This implies that setting the minimum number of polygons of a model to be between 1/16 and 1/64 of the original can bring performance gain while still maintaining the visual rendering quality. A similar optimization strategy can be used for other VEs.

8 CONCLUSION AND FUTURE WORK

We proposed a computational framework for tracking visually attended objects in interactive VEs, which utilizes the bottom-up preattentive features such as luminance, hue, depth, size, and motion and the newly included top-down contextual information such as spatial and temporal intention inferred from the user’s navigation patterns. The implemented framework was accelerated using the fragment shader in GPU, and it can render as high as 256 saliency maps in real time. The prediction accuracy of our framework was evaluated in a user experiment, and the results confirmed the critical role of top-down contexts in attention prediction. As a practical application, we demonstrated an effective and perceptually improved LOD management with our attention tracking framework, producing usual performance regulation capability and low cost-to-benefit ratios.

In the future, we will work on improving the spatial context model to include the novelty of objects of consideration, and developing adequate top-down context models for other common user tasks in VEs. We will then apply the attention tracking framework to depth-of-field rendering, preferably in more immersive display systems such as the CAVE.

ACKNOWLEDGMENTS

The authors would like to thank the four reviewers for their insightful comments and Toru Miyazawa at Toucan Co. for granting use of the fish models. This work was supported in parts by the Korea Science and Engineering Foundation (KOSEF) grant funded by the Korea government (MOST) (R01-2006-000-11142-0, R01-2006-000-10808-0, and R0A-2008-000-20087-0). Correspondence concerning this article should be addressed to Gerard J. Kim.

REFERENCES


Sungkil Lee received the BS degree in materials science and engineering at POSTECH, Korea, in 2002. He is a PhD candidate in computer science and engineering at POSTECH. His main research interests include virtual reality and real-time computer graphics based on visual and spatial perception. He is a student member of the IEEE and the IEEE Computer Society.

Gerard Jounghyun Kim received a bachelor’s degree in electrical and computer engineering at the Carnegie Mellon University in 1987 and a master’s degree in computer science at the University of Southern California in 1989 and 1994, respectively. He is currently an associate professor in computer science and engineering at Korea University. Prior to joining Korea University, he has spent 10 years directing the Virtual Reality Laboratory, POSTECH, Korea. His main research interests include various topics in virtual and mixed reality, human computer interaction (3D interaction and innovative applications, presence, VR/AR authoring, and experience-based computing), and computer music. He is a member of the IEEE.

Seungmoon Choi received the BS and MS degrees in control and instrumentation engineering at Seoul National University in 1995 and 1997, respectively, and the PhD degree in electrical and computer engineering at Purdue University in 2003. At present, he is an assistant professor in the Department of Computer Science and Engineering at POSTECH, Korea, directing the Haptics and Virtual Reality Laboratory. His main research interests include haptics and its application to virtual reality and robotics and applied perception. He is a member of the IEEE and the IEEE Computer Society.

For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/publications/dlib.