가상환경에서의 관심 추적 및
실시간 지각 기반 렌더링

2009

이성길

박사학위논문

가상환경에서의 관심 추적 및 실시간 지각 기반 렌더링

이 성 길(李成吉)

전자컴퓨터공학부(컴퓨터공학 전공)

포항공과대학교 대학원

2009

가상환경에서의 관심 추적 및 실시간 지각 기반 렌더링

Real-Time Perceptual Rendering with Computational Visual Attention Tracking in Virtual Environments

Real-Time Perceptual Rendering with Computational Visual Attention Tracking in Virtual Environments

by

Sungkil Lee

Division of Electrical and Computer Engineering
(Computer Science and Engineering)
POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY

A thesis submitted to the faculty of Pohang University of Science and Technology in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Division of Electrical and Computer Enginneering (Computer Science and Engineering)

Pohang, Korea
November 20, 2008
Approved by

Seungmoon Choi, Major Advisor

가상환경에서의 관심 추적 및 실시간 지각 기반 렌더링

이성길

위 논문은 포항공대 대학원 박사 학위 논문으로 학위 논문 심사 위원회를 통과하였음을 인정합니다.

2008 년 11 월 20 일

학위논문 심사위원회 위원장	최 승 문 (인)
위 원	김 정 현 (인)
위 원	이 승 용 (인)
위 원	최 승 진 (인)
위 원	이 인 권 (인)

DECE 20022231 이 성 길, Sungkil Lee, Real-Time Perceptual Rendering with Computational Visual Attention Tracking in Virtual Environments. 가상환경에서의 관심 추적 및 실시간 지각 기반 렌더링, Division of Electrical and Computer Engineering (Computer Science and Engineering), 2009, 106 P, Advisor: Seungmoon Choi. Text in English

Abstract

This dissertation presents a real-time perceptual rendering framework based on computational visual attention tracking in a virtual environment (VE). The visual attention tracking identifies the most plausibly attended objects using top-down (goal-driven) contexts inferred from a user's navigation behaviors as well as a conventional bottom-up (feature-driven) saliency map. A human experiment was conducted to evaluate the prediction accuracy of the framework by comparing objects regarded as attended to with human gazes collected with an eye tracker. The experimental results indicate that the accuracy is in the level well supported by human cognition theories. The attention tracking framework, then, is applied to depth-of-field (DOF) rendering and level-of-detail (LOD) management, which are representative techniques to improve perceptual quality and rendering performance, respectively. Prior to applying the attention tracking to DOF rendering, we propose two GPU-based real-time DOF rendering methods, since there have been few methods plausible for interactive VEs. One method extends the previous mipmap-based approach, and the other, the previous layered and scatter approaches. Both DOF rendering methods achieve real-time performance without major artifacts present in previous methods. With the DOF rendering methods, we demonstrate attention-guided DOF rendering and LOD management, which use the depths and the levels of attention of attended objects as focal depths and fidelity levels, respectively. The attention-guided DOF rendering can simulate an interactive lens blur effect without an eye tracker, and the attention-guided LOD management can significantly improve rendering performance with little perceptual degradation.

Contents

1 Introduction **1**

1.1 Computational Visual Attention Tracking 2

1.2 Depth-of-Field Rendering 3

1.3 Level-of-Detail Management 4

1.4 Contributions 4

1.5 Organization 5

2 Background and Previous Work **6**

2.1 Previous Approaches for Visual Attention Estimation 6

2.2 Depth-of-Field Rendering 8

2.2.1 Multipass Approach 8

2.2.2 Post-Filtering Approach Using Single Layer 9

2.2.3 Post-Filtering Approach Using Multiple Layers 10

2.2.4 Bokeh Rendering 11

2.3 LOD Management 12

3 Real-Time Tracking of Visually Attended Objects **13**

3.1 Real-time Bottom-Up Saliency Map 14

3.1.1 Bottom-Up Features and their Image Pyramids 15

3.1.2 Pixel-Level Saliency Map 16
3.2 Modulation by Top-Down Contexts 17
3.2.1 Object-Level Attention Map 17
3.2.2 Spatial Context Based on User Motion 18
3.2.3 Temporal Context 19
3.3 Implementation Details 20
3.4 Computational Performance 22
3.5 Evaluation of Attention Prediction Accuracy 23
3.5.1 Methods 23
3.5.2 Data Analysis 26
3.5.3 Results 27
3.5.4 Discussion 30

4 Real-Time Depth-of-Field Rendering Using Anisotropically Filtered Mipmap Interpolation 33
4.1 DOF Rendering Based on Nonlinear Mipmap Interpolation 36
4.1.1 Construction of Mipmap Pyramid 36
4.1.2 Computing CoC from Depth 37
4.1.3 Relating CoC to Mipmap Level 38
4.1.4 Nonlinear Mipmap Interpolation with Circular Filtering 39
4.2 Anisotropic Mipmap Construction for Intensity Leakage Reduction 40
4.3 Smoothing Foreground Boundaries 43
4.4 Implementation Details 45
4.5 Results 47
4.5.1 Rendering Quality 47
4.5.2 Rendering Performance 51
4.6 Discussion 53

5 Real-Time Depth of Field Rendering using Point Splatting on Per-Pixel Layers 55

5.1 Depth-of-Field Rendering Algorithm 57

5.1.1 Relating Depth to CoC 58

5.1.2 Rendering of Partially Occluded Area 58

5.1.3 Image-Space Point Splatting 60

5.1.4 Splatting on Per-Pixel Layers 61

5.1.5 Composition of Layers and Normalization 63

5.2 Defocused Highlights 64

5.3 Acceleration of Rendering Performance 65

5.4 Results and Discussion 67

6 Attention-Guided Depth-of-Field Rendering 71

6.1 Integration of Attention Tracking to Depth-of-Field Rendering 71

6.2 Performance Evaluation 74

6.3 Discussions 76

7 Attention-Guided LOD Management 78

7.1 LOD Management Using Attention Estimation Framework 78

7.2 Cost Model of Attention-Guided LOD Management 79

7.3 Associating ULOD with Object Attention Values 82

7.4 Performance Evaluation 83

7.5 Discussion 85

8 Conclusion and Future Work 88

한글 요약문 **90**

Bibliography 91

List of Figures

3.1 The overall procedure for the proposed visual attention tracking. 14

3.2 An example of the spatial context model. An observer is currently moving toward the target object ($\mathbf{v} \cdot \mathbf{w} > 0$). The left fish moving in the opposite direction to navigation ($\Delta x > 0$) is ignored for top-down contextual modulation. The other fish is under the influence of the distance emphasis model (Weibull curve in the right) and the screen emphasis model (the 2D Gaussian weights on the image plane). 19

3.3 An excerpt from the GLSL fragment shader program—the center-surround difference operation for the DSM texture image. 21

3.4 Examples of results using the visual attention tracking framework, for static (upper) and dynamic (lower) scenes. All the images were rendered in real-time. In the static scene, the motion feature was not used. The whitest object (the bowling pin in the upper image and the orange fish in the lower image, respectively) is the most attentive object in the corresponding object attention map (S_o). The 3D fish models were provided through the courtesy of Toru Miyazawa at Toucan Co. 22

3.5 Comparison of measured computation times for generating a saliency map, between the CPU-based and GPU-based (ours and Longhurst et al.'s [60]) implementations. 23

3.6 A monocular eye tracker (Arrington Research, Inc.) and a 42-inch LCD display used in the experiment. 24

3.7 The virtual undersea (dynamic; upper row) and virtual gallery (static; lower row) environments used in the experiment. 25

3.8 Two examples of number-attached objects. For fishes in the virtual undersea, tags were attached to both sides of the abdomen. For the virtual gallery, they were four tags around the middle section of an object. 25

3.9 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. 28

3.10 Means and standard errors of the accuracies with and without B, E, and T for free navigation (upper) and visual search task (lower), respectively. . . . 29

4.1 Example images generated with our DOF rendering method and their depth maps (in the bottom-left corner of each image). The three scenes except the "Forest" scene (at the bottom right) were rendered in real time. The "Forest" scene was postprocessed in real time from an offline rendered image. Our method generates realistic DOF effects without any noticeable bilinear magnification artifacts, intensity leakage, and blurring discontinuity. 34

4.2 The thin-lens model [81]. 37

4.3 A mapping from mipmap levels to the standard deviations of a Gaussian filter that produces the most similar images in terms of RMSE. 38

4.4 A comparison between (a) rectangular sampling and (b) our circular sampling for Gaussian filtering. In the circular sampling, diagonal samples (red points) are located more closely toward the center. 40

4.5 A comparison of DOF effects with (a) bilinear magnification and (b) our magnification method with finer mipmap images and the circular blending. The bilinear magnification yields severe jagged artifacts, while ours almost removes the artifacts and well maintains the object shapes. 40

4.6 An illustration of the two 3×3 Gaussian filters used during mipmapping. The elevated pixels represent sharp pixels (lower σ than other pixels), and the arrows indicate the direction of intensity flow. 41

4.7 A comparison of the results generated by the isotropic and anisotropic Gaussian mipmapping. The intensity leakage around the sharp area (leaked brown colors in the left image) was greatly reduced in the right image. . . . 42

4.8 DoB and DoN textures: (a) an original DoB image, (b) a DoB image foreground-smoothed using $\rho(M)$, and (c) a DoN image read from the highest mipmap level, M. Note that the background pixels in $\rho(M)$ are black (i.e., $\rho(M)$=0). The images were scaled for illustration. 44

4.9 A comparison of the results generated with (a) discontinuous DoBs and (b) smoothed DoBs using our method. The blurring discontinuities are significantly reduced in the right image. 45

4.10 The overall framework of our DOF rendering applied at every rendering frame. 46

4.11 A comparison of background-blurred images rendered by our method and three recent GPU-based methods [87, 103, 53]. The nearest can is focused. The reference image was rendered by the accumulation buffer method [43], and used for computing $PSNR$s. 47

4.12 A comparison of foreground-blurred images rendered by the same methods in Fig. 4.11. The farthest wall is focused. 49

4.13 The evolution of blurred appearance. When the mean CoC diameter ($\bar{C}$) is equal to or less than 16 pixels, the artifacts do not stand out. 50

4.14 Performance comparison in terms of frame rate for the four methods: ours, Zhou et al. [103], Scheuermann [87], and Kraus et al. [53]. 52

5.1 Two example images generated by our DOF rendering method. The upper image shows no intensity leakage, and the lower image shows a partial occlusion effect with defocused highlights of a hexagonal aperture. 56
5.2 Overview of our DOF rendering framework. 57
5.3 Smooth intensity decrease in the splatted image using CoCs of a 24-pixel diameter for the boundary of the white rectangle. The distance from the edge to the inner area of the full intensity is roughly same to the radius of the CoC. 59
5.4 Accepted pixels after the first and second depth tests (left), and the third test (middle) for the Elephant scene (right). The farthest zebra is focused. . . . 59
5.5 Example rasterization patterns and scaling constants in terms of a point size for the uniform IDF. 61
5.6 An example of the layer decomposition. 62
5.7 Example images blended without (left) and with (right) normalization. Darker and brighter regions in the left image were correctly normalized in the right image. 63
5.8 Example results with and without defocused highlights rendered with HDR and LDR images. The bottom-left portion in each image represents the IDF texture used (hexagonal and Gaussian IDFs). Each highlighted image shows better intensity contrast. 65
5.9 Rendered images at the original (left) and reduced (right) resolutions. Normalization was not applied. 66
5.10 A comparison of DOF rendering results by our method and those of [103] and [43]. The rounded boxes represent the magnified portions in the images for comparison of foreground blurring. 68

5.11 Example images rendered for a scene focused on the near (top left), middle, (top right), and far (bottom left) objects. Whereas the first two are well represented, the third exhibits small artifacts (see the red and green boxes magnified in the bottom right side). 69
5.12 Two example images generated using per-pixel layers (left) and globally defined layers (right) for a scene with long objects. 70

6.1 Examples of saccades simulation using the linear interpolation and lowpass filter. 73
6.2 A test scene used for simulating saccadic eye movements. The focus was altered between the elephant and zebra. 74
6.3 Comparison of measured frame rates for rendering DOF effects with the visual attention tracking by the methods presented in Chapters 4 and 5. . . . 75

7.1 Comparison between the object attention maps rendered using the raw and simplified models. The upper images are rendered color images, and the lower images, the corresponding object attention maps. The triangle numbers of simplified models were 1/4, 1/16, 1/64, and 1/256 of the original one, respectively. The Bunny model was provided through the courtesy of Stanford 3D Scanning Repository. 81
7.2 Simplified Bunny models for LOD management. The original model consists of 69,451 triangles. The triangle numbers of the simplified models decrease by a quarter scale from left to right (69,451, 17,364, 4,341, 1,086 and 272 triangles, respectively). 83
7.3 Two example images rendered without (upper) and with (lower) LOD management. In the lower image, non-attentive objects were simplified based on their object attention values. The Dragon model was provided through the courtesy of Stanford 3D Scanning Repository. 84

7.4 Mean rendering costs under NOLOD, ULOD2, ULOD3, ULOD4, ULOD5 in the Bunny and Dragon scenes. The vertical error bars represent the standard errors. 85
7.5 Evolution of the rendering costs with respect to the number of visible objects in the Bunny (left) and Dragon (right) scenes. The triangle number of a non-simplified Bunny was 69,541. The triangle number of a Dragon was 435,708. 86

List of Tables

3.1 Source table of the within-subject 2×2×2 ANOVA, for the effects of task (K) and environment (V). 28

3.2 Means and standard errors of the estimation accuracies for the eight combinations of feature groups (B, E, and T). 30

4.1 Examples for the number of texture lookup operations required in GPU programs for the four methods. The largest CoC is 16 pixels ($\sigma_{max} = 8$) at a 1024×1024 display resolution. 1,048,576 ($= 1024 \times 1024$) operations are considered one unit. 53

5.1 A performance benchmark for the four methods in terms of frame rate for Elephant (E) and Grasshopper (G) scenes. Subscripts represent mean CoC sizes. 67

Chapter 1

Introduction

Perceptual rendering in a 3D virtual environment (VE) can be largely categorized into selective rendering techniques to yield better perceptual effects or lower computational costs. For instance, the former includes depth-of-field (DOF) rendering [81, 84, 72], peripheral degradation (or foveation) [95], and stereoscopy with correct convergence. The latter includes level-of-detail (LOD) management [16, 13], geometry simplification [56], information culling in a distributed VE [9], and global illumination [102, 60]. To create a perceptually effective VE, these two mainstreams commonly require identifying a region of interest (ROI) and plausible rendering techniques for interactive VEs. However, there have been few practical methods to identify a ROI in interactive VEs beyond the traditional feature-driven methodologies for images or movies. Furthermore, in contrast to the attempts for lower computational costs (by degrading less perceptible objects), perceptual quality improvements for identified ROIs have been relatively little explored. In this dissertation, the author proposes a general rendering framework to bring a perceptual benefit to the user of an interactive VE. With a computational framework to track a ROI (or visual attended objects) in a VE and real-time high-quality DOF rendering techniques, the author demonstrates two representatives of perceptually enhanced and efficient rendering applications: *attention-guided DOF rendering* and *LOD management*.

1.1 Computational Visual Attention Tracking

Visual information transfer has been known to be selectively modulated by *visual attention* [12, 94]. 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 interactive 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.

Inspired by the feature integration theory of Treisman and her colleagues [94], a number of approaches have been proposed to computationally estimate visual attention in the retinal image [51, 23, 71, 47, 5]. However, the majority of them have only considered *pixel*-based regions rather than *objects*, whereas visual attention changes in association with the movement of objects rather than location-based movements [28, 75, 89]. Thus, their methods cannot guarantee temporal coherency during tracking attended objects. Moreover, the ROI derived from their framework based on the conventional bottom-up (*stimulus-driven*) features may incorrectly predict attention, since their framework does not reflect a user's volitional factors as reported in the previous literature [58, 99, 44]. In addition, the cost for computing level of visual attention has been too expensive to be used in real time. Due to these limitations of the previous approaches, computational attention models adequate for dynamic VEs have received relatively little attention.

In this dissertation, the author presents a real-time computational framework for predictive tracking of visual attention in dynamic and interactive VEs. Our framework allows smooth tracking of visually attended objects over period of time, and employs user's intention inferred from a user's spatial and temporal behavior during navigation in a dynamic VE. Our framework was implemented using the recent graphics processing unit (GPU), enabling a remarkable real-time performance that is fast enough to be used for interactive VEs. We also conducted a user experiment using an eye tracker to evaluate the accuracy of our attention prediction.

1.2 Depth-of-Field Rendering

DOF represents a distance range around a focal plane, where objects appear to be in the focus for a camera or human visual system [41, 33, 59]. A finite aperture of a lens maps a 3D point to a circular region in an image, called the *circle of confusion* (CoC) whose size is proportional to its distance to the focal plane. Overlapped CoCs in the image make defocused objects blurred, while focused objects within the DOF are seen sharply. In general, the DOF effect is known to improve photorealism, to mediate the monocular depth perception [64, 68], and to make a focused object distinctive from the background.

As can be seen the definition of the DOF, the simulation of the DOF effects essentially requires identifying the depth of a focused object. Although the focus can be found using an eye tracker, limited movements of a user caused by the eye tracking have restricted its practical use in VR applications, except for few desktop applications [45]. As already alluded to, our attention tracking framework is a feasible alternative to the eye tracking, allowing the interactive DOF rendering without any restrictions on user's movements. Attention-directed DOF rendering in interactive VEs can be realized simply using the depth of the object predicted to be attended to as a focal depth. The combined DOF rendering system takes the aforementioned benefits.

However, in addition to the identification of a focus, another problem arises from the lack of feasible rendering methods for simulating DOF effects. Unlike the real-lens system with a finite aperture, conventional computer graphics systems use a pinhole lens model, which requires additional costly image processing for DOF effects. A number of rendering algorithms have been developed in the past, forming a spectrum of quality-performance tradeoff, from accurate to fast approaches (see Section 2.2 for a detailed review). Nonetheless, even the state-of-the-art techniques have been still insufficient to be used for VR applications in terms of either image quality or rendering performance.

Here, the author proposes two real-time DOF rendering methods sufficient for our purpose, which intensively exploit capabilities of the modern GPU. The first method uses a mipmap interpolation for remarkable real-time performance (see Chapter 4 for details), and also considerably reduces major visual artifacts commonly present in the previous techniques. The other method uses point splatting on per-pixel layers (see also Chapter 5 for details), which produces the images similar to those of the previous accurate (but very slow) techniques [22, 43].

1.3 Level-of-Detail Management

Despite the recent strides in computer graphics and VR, the designer of a VR system still struggles with the trade-off between complexity and performance. The resolution of this classic issue is usually attempted by managing LOD of objects in a VE. The most important aspect in the LOD management is the decision logic for switching between different model resolutions during runtime rendering. Staring from the simple metric based on the distance of objects from an observer, a number of approaches have been proposed (see Section 2.3 for more details). However, their perceptual degradation is easily noticeable, since they are weakly related to the human visual attention, in particular, for the user's volitional shifts. Except 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.

In this dissertation, we demonstrate how to apply our framework to LOD management using "Unpopping LOD" (ULOD) [38] that is a recent image-space blending technique for coping with the "popping" effect during the discrete LOD switching. 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. The combined rendering system allows greater computational performance with little degradation in the perceptual quality.

1.4 Contributions

The major contributions of this dissertation is summarized as follows:

- development and experimental evaluation of a visual attention estimation framework plausible for real-time VR applications, which allows object-level coherent tracking and involves user's intention for more accurate prediction,
- development of the two real-time DOF rendering methods, which are applicable to VR applications in terms of remarkable rendering performance and image quality,
- proposal of the DOF rendering scheme combined with the visual attention tracking, and
- demonstration of the LOD management scheme that combines the visual attention tracking with ULOD.

1.5 Organization

The remainder of this dissertation is laid out as follows. Chapter 2 reviews background and previous work related to visual attention estimation, DOF rendering, and LOD management, respectively. Chapter 3 describes how visually attended objects are predicted and tracked using bottom-up saliency and top-down contextual modulation. We also explain how the visual attention tracking framework is efficiently implemented using the GPU, and experimentally evaluate the accuracy of the attention prediction. Chapters 4 and 5 present the two real-time methods for rendering DOF effects in VR applications. Chapter 6 deals with how DOF rendering is integrated with visual attention tracking. In Chapter 7, the LOD management associated with the visual attention tracking is demonstrated and its performance gain is analyzed. Finally, Chapter 8 concludes this dissertation.

Chapter 2

Background and Previous Work

2.1 Previous Approaches for Visual Attention Estimation

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 [44, 20]. Color and intensity in images are common examples of bottom-up stimuli [94]. Prior knowledge, memories, and goals can be top-down factors [58, 99]. 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 [94] 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*.

Inspired by the feature integration theory, several computational models have been proposed to estimate visual saliency in an image [51, 23, 71, 47, 5]. In particular, the model of Itti et al. [47] 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 [51]. 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 [47] or the difference-of-Gaussian filter [48]. This framework has been applied and extended to various applications including object recognition [85], video compression [77], and video summarization [63]. The bottom-up component of our framework extends the model of Itti et al. using two image features (luminance and hue) and three 3D dynamic features (depth, object size, and object motion).

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 [101, 35, 99, 44]. 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 [30], 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.

Since images or movies used in the previous studies do not allow user's interaction, unlike interactive VR applications, the effects of the user's intention for predicting attention have been considered using user-defined importance, image features, or eye tracking. Cater et al. introduced the importance factor of objects relative to a task in computing saliency [16], but the factor values had to be set manually. Features such as object distance from a viewer, image coverage [42], and novelty [102, 60] 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. [80] demonstrated the effect of task-dependent influences in video games, using a learning methodology based on the bottom-up features. However, their consideration of top-down influence is related to the specific task or environment, and thus, prediction of the individual intention of a user does not seem straightforward. In our framework, to effectively infer intention of individuals in a VE, we use spatial configuration of objects from the observer (we call this value the *spatial context*), which are obtained from atomic interaction data during spatial navigation in a VE. The temporal trends of this spatial context is further tracked via linear filtering. The spatial and temporal contexts are used to select the most plausibly attended objects among the bottom-up salient candidates.

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. [78] compared the saliency model of Itti et al. [47] 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 non-photorealistic scenes and compared its prediction performance with that of an eye tracker [86]. However, compared to the abundant theories and applications of the saliency map, efforts toward their formal validation have been somewhat insufficient. In our work, we conducted a human 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.

2.2 Depth-of-Field Rendering

DOF rendering was pioneered by Potmesil et al. [81] and since then, has been followed by a number of algorithms. We categorize them into three groups and provide brief reviews for each. We also describe the previous approach for rendering "bokeh" effects. Our first DOF rendering method (Chapter 4) extends the previous post-filtering method based on the gather, and the other method (Chapter 5) efficiently combines the previous post-filtering method based on the scatter and layer composition.

2.2.1 Multipass Approach

The most accurate approach to render DOF effects is simulating multiple rays within the lens with a finite aperture. The distributed ray tracing [22] and the accumulation buffer method [43] are such examples. They can handle shading from various rays and partial occlusion (visibility of an object varies with different sampling positions on a lens), but they usually require a large number of view samples (i.e., the number of model rendering) proportional to the size of the CoC for producing a convincing DOF effects. Due to the heavy computational cost, the multipass methods are generally considered inappropriate for real-time applications. A notable recent advance in this approach is a technique called the "time-continuous triangles" [2] that is extended from the accumulation buffer method using a reduced number of samples for improved rendering speed. However, the work is still conceptual and requires considerable modifications of the existing graphics hardware to be implemented.

2.2.2 Post-Filtering Approach Using Single Layer

Another main stream to DOF rendering has been the post-filtering approach. This approach computes the blurring degree (i.e., the CoC diameter) for each pixel depth using the thin-lens model proposed by Potmesil et al. [81], except the pre-blurring methods which use a crude linear approximation. The post-filtering approach using a single layer can be classified into the scatter and gather methods depending on how an image is blurred. Those using multiple layers are discussed in Section 2.2.3. Our two DOF rendering methods were inspired by the gather and scatter approach, respectively. Each of them improves the previous approaches in terms of rendering performance and image quality.

The scatter method [81, 90] distributes the intensities of source pixels in a pinhole image onto circular sprites (i.e., CoCs) for blurring. Then, the sprites are blended from farther to nearer, and normalized such that the accumulated intensity for each pixel equals to one. Since sharp pixels are not scattered to the neighborhood, intensity leakage is absent inherently. This method, however, requires heavy sorting of entire depths, and thus, it has been used in offline software. Our second method extends this scatter method with per-pixel layer composition to improve rendering performance.

The gather method, on the other hand, simulates blurring of pixels by spatially convolving (filtering) neighbor pixels within the CoC of a pixel. Starting from iterative filtering with a small kernel [84], algorithms have evolved to a number of techniques including the pre-blurring [26, 83, 87, 39, 29], the anisotropic diffusion [10], the ring-shaped kernel [72], and the separable Gaussian filtering [83, 103]. Most of them intensively utilize the texture lookup capability of recent GPUs, and hence, they can achieve real-time performance.

However, most gather techniques involve two severe visual artifacts: intensity leakage and blurring discontinuity. The intensity leakage results from the spatial filtering that does not account for the CoC variations around a pixel. Among them, Bertalmío et al. has successfully alleviated the leakage using the anisotropic diffusion [10]. However, their computational complexity significantly increases with a CoC size, which is undesirable for interactive applications. In contrast, the pre-blurring techniques achieve superior real-time performance due to their simplicity. However, a naively downsampled image contains the leakage inevitably, and the lack of intermediate blurring information—because only one down-sampled image is used—severely deteriorates the image quality. In order to remove intensity leakage, our first DOF rendering method combines the anisotropic diffusion and the pre-blurring methods.

As for the blurring discontinuity, most of post-filtering methods (using a single layer) have not addressed the issue due to the lack of directional information from different lens spots. A notable remedy is the work of Bertalmío et al. [10] that uses a blurred depth buffer. However, their method sets the blurring degree (corresponding to a CoC size) to be zero around regions where the foreground and background meet. Furthermore, the depths around a non-occluded focused region are blurred. In contrast, our first method using mipmap framework reduces the artifacts by smoothing the degrees of blurring instead of depths.

2.2.3 Post-Filtering Approach Using Multiple Layers

The post-filtering approach using multiple layers has been developed to cope with the two problems of the intensity leakage and the lack of partial occlusion. Inspired by an earlier object-space grouping method [88], several layer composition methods have been proposed [8, 7, 49, 52, 53]. They decompose a pin-hole image into several sub-images according to the depth of pixels. The sub-images are separately blurred using the Fast Fourier Transform [8, 7], a customized pyramidal processing [53], or anisotropic heat diffusion [49, 52], and then blended from farther to nearer depths (similarly to the scatter method).

This approach produces better results compared to the single-layer methods in terms of intensity leakage and foreground blurring quality. However, they usually require sufficient layers for generating natural blurring transitions between layers, and are inappropriate for real-time VR applications (e.g., 156 seconds with 11 depth slices for [8], 14.2 ms with 12 sub-images for [53], and 6–7 frames/sec for [49]). Furthermore, the discretization problem (an object being separated across more than one layers) [6] can still degrade the image quality in areas where layers are overlapped, in spite of their remedies for the problem [8, 6, 53]. In comparison, our second method adaptively decomposes layers using local depth variation, and the discretization artifact rarely occurs unless different objects are located too close to each other.

In the previous multi-layer approach, partial occlusion is commonly addressed by extrapolating the exterior boundaries of invisible areas. However, the area filled by their methods behind foreground layers represents the blurred color of boundaries rather than the accurate (possibly sharp) colors of the occluded area. This artifact becomes more apparent with a large CoC that cannot be covered by mere extrapolation. In particular, Kraus et al. [53] used pyramidal processing similar to our mipmap approach (the first method). Their image quality for the foreground blurring can be better than ours for a static image, but may exhibit some flickering in an animated scene due to the use of significantly blurred images for filling occluded areas. In contrast to the all previous methods including our first method, our second method effectively takes care of this artifact with an additional rendering of the occluded area.

2.2.4 Bokeh Rendering

“Bokeh” refers to the blurred appearance distorted by the aperture shape and *intensity distribution function* (IDF) of a lens. For bokeh rendering, previous studies proposed a stochastic sampling scheme for the multi-pass rendering [14] and a gather method [83]. However, implementing the correct bokeh with the gather is rather difficult (e.g., [49]), since bokeh patterns are determined from the scattered pixels of a source rather than the neighbor pixels to be gathered. Our second method follows the scatter and can simulate the accurate bokeh effects, while our first method cannot simulate point spread functions other than Gaussian IDF.

2.3 LOD Management

From an early multi-resolution representation for a single object (later called the discrete LOD or DLOD) [19], techniques for LOD management have evolved into numerous continuous and view-dependent methods (e.g., see [46, 62, 100]). As already mentioned, the most important aspect in LOD management is the decision logic for switching between different model resolutions. 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 [61], 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 [36] or by using relatively simple information such as user-defined importance and a bottom-up saliency model [102]. 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 [61]. Since all of these measures are closely related to the visual attention, our attention tracking framework allows the effective LOD management framework, which are highly correlated with the previous heuristics. This has not been possible without an eye-tracking device prior to our attention tracking framework.

Chapter 3

Real-Time Tracking of Visually Attended Objects

The overall procedural flow of our framework for visual attention tracking is summarized in Fig. 3.1. 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. [47] 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. 3.1). Then, the image pyramids are converted by the center-surround difference to build contrast maps (alternatively called the conspicuity maps in [47]) for each feature (step 2 in Fig. 3.1), 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. 3.1).

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. 3.1). The relevance of an object to a given task is also considered in

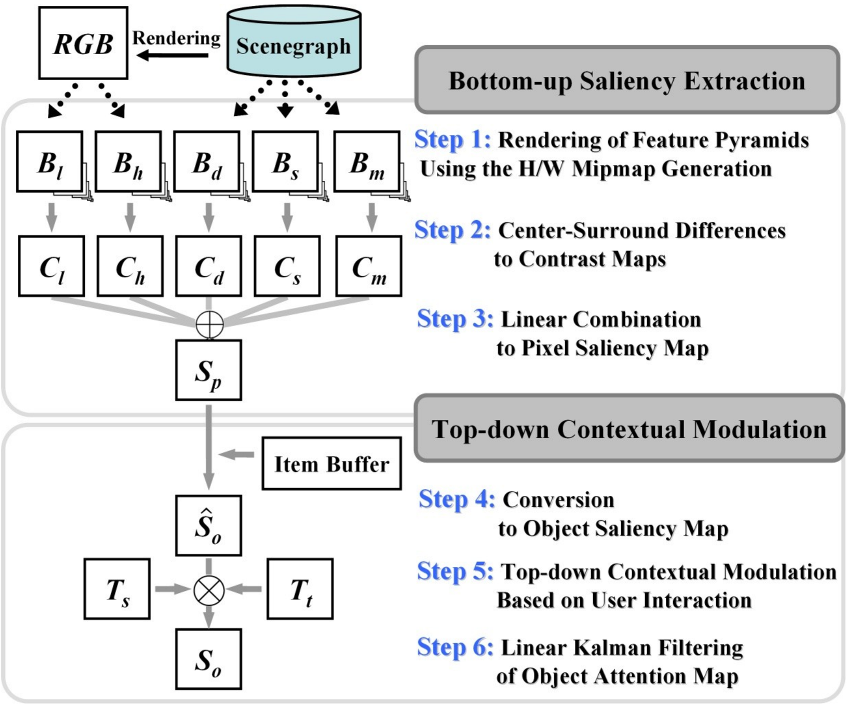

RGB
Rendering
Scenegraph
Bottom-up Saliency Extraction
B_l
B_h
B_d
B_s
B_m
Step 1: Rendering of Feature Pyramids Using the H/W Mipmap Generation
C_l
C_h
C_d
C_s
C_m
Step 2: Center-Surround Differences to Contrast Maps
Step 3: Linear Combination to Pixel Saliency Map
S_p
Top-down Contextual Modulation
Item Buffer
$\hat{S}_o$
Step 4: Conversion to Object Saliency Map
T_s
T_t
Step 5: Top-down Contextual Modulation Based on User Interaction
S_o
Step 6: Linear Kalman Filtering of Object Attention Map

Fig. 3.1 The overall procedure for the proposed visual attention tracking.

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. 3.1). 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. 3.1). In subsequent sections, detailed explanations for each step are provided.

3.1 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.

3.1.1 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 [94] for luminance, [73] for hue, [31] for depth, [94] for size, and [74] for motion). Compared to the original image features used by Itti et al. [47], 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 HLS (Hue-Luminance-Saturation) 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:

$$B_d = \frac{z - z_{near}}{z_{far} - z_{near}}, \tag{3.1}$$

where z, z_{near}, and z_{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:

$$B_s(k) = \frac{\text{number of pixels belonging to object } k}{\text{total number of pixels in the image}}. \tag{3.2}$$

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 [97].

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

$$B_m(k) = \|\mathbf{p}^{\tau}(k) - \mathbf{p}^{\tau-1}(k)\|, \tag{3.3}$$

where **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 4.4 for more details).

3.1.2 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 [47]. This is an operation that detects locations standing out from their surroundings. The center-surround difference for a contrast map, C_f ($f \in \{l, h, d, s, m\}$), is calculated as:

$$C_f = \frac{1}{6} \sum_{i \in \{0,1,2\}} \sum_{\mathrm{J} \in \{3,4\}} |B_f^i - B_f^{i+j}|, \tag{3.4}$$

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 [47], and is widely used for bottom-up saliency map computation.

The contrast maps are merged into a single topographical saliency map, $\bar{S_p}$, by linearly combining them as:

$$\bar{S_p} = \sum_{f \in \{l,h,d,s,m\}} w_f C_f, \tag{3.5}$$

where w_f's are linear combination weights. Even though various schemes can be used for determining the weights [47], most of them are not suitable for real-time use. In our framework, the weights are set to:

$$w_f = \frac{1}{\max_{(u,v)} C_f(u, v)}, \tag{3.6}$$

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, $\bar{S_p}$ is normalized to S_p (the pixel-level saliency map), such that each pixel value in S_p ranges in $[0, 1]$.

3.2 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.

3.2.1 Object-Level Attention Map

We first convert the pixel-level saliency map to an object-level saliency map ($\hat{S_o}$). As noted in [75, 89], 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:

$$\hat{S_o}(k) = T_i(k)\frac{1}{n(k)} \sum_{(u,v)\in \text{object } k} S_p(u, v), \tag{3.7}$$

where $n(k)$ is the number of pixels of 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 [97]. The task-related 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 and 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:

$$S_o(k) = (T_s(k) + T_t(k))\hat{S_o}(k). \tag{3.8}$$

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 [98].

3.2.2 Spatial Context Based on User Motion

Typical models of spatial perception such as the topological 3-stage model (landmark, route, and survey knowledge) require a hierarchical cognitive map of objects [91, 54]. 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 et al. [24], 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, $\mathbf{v}$ be the viewing direction of the observer, and $\mathbf{w}$ be the moving direction of the observer (see Fig. 3.2 for conceptual illustrations). $\Delta x = x^{\tau} - x^{\tau-1}$ is the difference in x between two consecutive simulation frames, where $\tau - 1$ and τ are the corresponding time indices. Firstly, 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: $e^{-c_y y^2}$, where c_y is a scaling constant. Secondly, 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 $c_x = L/0.707$ is a scaling constant that is determined from a desired distance (L) of maximum spatial emphasis. Thirdly, when observers move

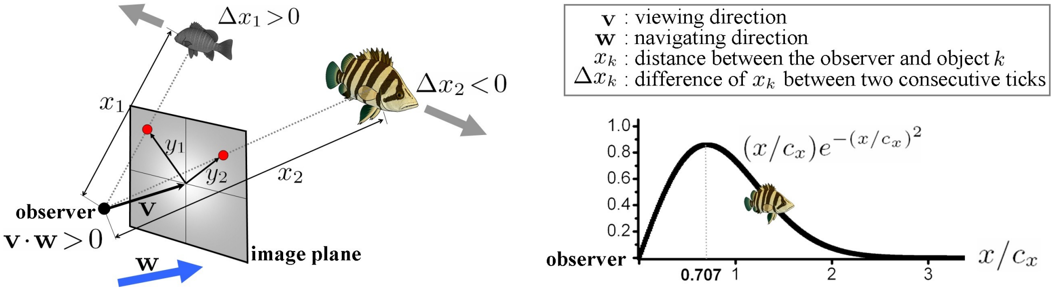

$\Delta x_1 > 0$
$\Delta x_2 < 0$
x_1
y_1
y_2
x_2
observer
$\mathbf{v}$
$\mathbf{v} \cdot \mathbf{w} > 0$
$\mathbf{w}$
image plane
$\mathbf{v}$: viewing direction
$\mathbf{w}$: navigating direction
x_k : distance between the observer and object k
Δx_k : difference of x_k between two consecutive ticks
1.0
0.8
0.6
0.4
0.2
$(x/c_x)e^{-(x/c_x)^2}$
observer
0.707
1
2
3
x/c_x

Fig. 3.2 An example of the spatial context model. An observer is currently moving toward the target object ($\mathbf{v} \cdot \mathbf{w} > 0$). The left fish moving in the opposite direction to navigation ($\Delta x > 0$) is ignored for top-down contextual modulation. The other fish is under the influence of the distance emphasis model (Weibull curve in the right) and the screen emphasis model (the 2D Gaussian weights on the image plane).

forward ($\mathbf{v} \cdot \mathbf{w} > 0$; also see Fig. 3.2), 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:

$$T_s(k) = \begin{cases} 0 & \text{if } \mathbf{v} \cdot \mathbf{w} > 0 \text{and } \Delta x > 0 \\ c_s(\frac{x}{c_x})e^{-(\frac{x}{c_x})^2} e^{-c_y y^2} & \text{otherwise} \end{cases}, \tag{3.9}$$

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

3.2.3 Temporal Context

High spatial context values observed for an object during a certain period of time implies 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, we define temporal context, $T_t(k)$, for object k using the running average of the spatial contexts as:

$$T_t(k) = \frac{1}{\lambda} \sum_{\tau=\tau_0-\lambda+1}^{\tau_0} T_s{}^{\tau}(k), \tag{3.10}$$

where λ denotes a time duration for controlling the long-term interest, and τ_0 and T_s^{τ} are the current simulation frame and the spatial context at τ, respectively. An additional role of the temporal context is to compensate for the erroneously derived spatial context values (e.g., unintended control errors).

3.3 Implementation Details

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 scenegraph, and a unique ID and task-related importance (T_i) were assigned to each.

During run-time, the computational procedure shown in Fig. 3.1 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 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. Firstly, 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 mipmaps using the hardware mipmap generation capability and compute the center-surround difference using the mipmaps. A sample fragment program for the center-surround differences is shown in Fig. 3.3. The texture look-up operation for a coarser scale corresponds to the magnification of an image, and thus it significantly improves computational performance, achieving

```
uniform sampler2D DSM;          // texture sampler
#define tc0 gl_TexCoord[0].st   // texture coordinate

void main()
{
  vec3 c[7];
  for(int i=0; i<7; i++)
    c[i] = texture2D(DSM,tc0,float(i)).rgb;
  vec3 C = abs(c[0]-c[3])+abs(c[0]-c[4])+abs(c[1]-c[4])+
           abs(c[1]-c[5])+abs(c[2]-c[5])+abs(c[2]-c[6]);
  gl_FragColor = vec4(C/6.0,1.0);
}
```

Fig. 3.3 An excerpt from the GLSL fragment shader program—the center-surround difference operation for the DSM texture image.

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 [97], which is an image where each pixel contains the ID of the object that the pixel belongs to. Using the ID image and the top-down contexts (T_i, T_s, 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 [98], based on the position and velocity state model.

Fig. 3.4 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, 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 and T_t), the most attentive objects are ultimately selected in a manner which is in greater correspondence with our

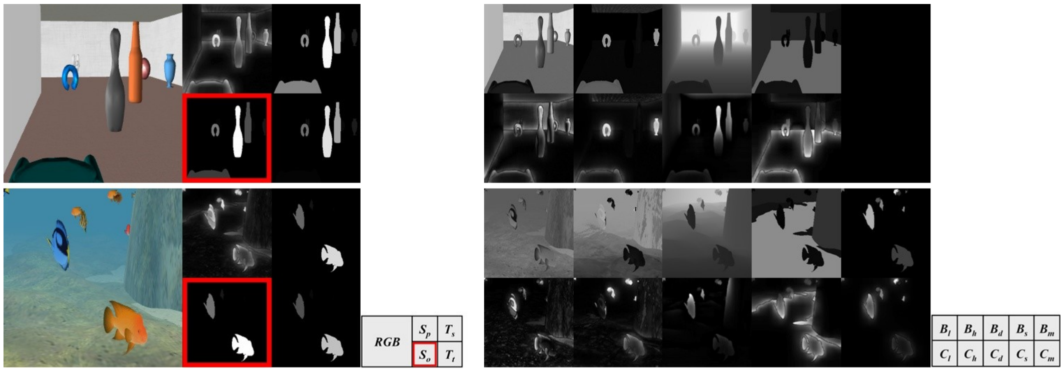


Fig. 3.4 Examples of results using the visual attention tracking framework, for static (upper) and dynamic (lower) scenes. All the images were rendered in real-time. In the static scene, the motion feature was not used. The whitest object (the bowling pin in the upper image and the orange fish in the lower image, respectively) is the most attentive object in the corresponding object attention map (S_o). The 3D fish models were provided through the courtesy of Toru Miyazawa at Toucan Co.

visual attention (to be validated in the next section).

3.4 Computational Performance

In order to compare the computational performance of our framework to a typical CPU-based method and the most recent GPU-based method [60], 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 model in Fig. 3.4). The comparisons were performed for four saliency map sizes (64×64, 128×128, 256×256, and 512×512). Fig. 3.5 shows the results of comparison. The speed-up compared to Longhurst et al.'s method is between 1.57 times (for the 64×64 image) and 1.15 times (for the 512×512 image). The speed-up compared to the CPU-based method is be-

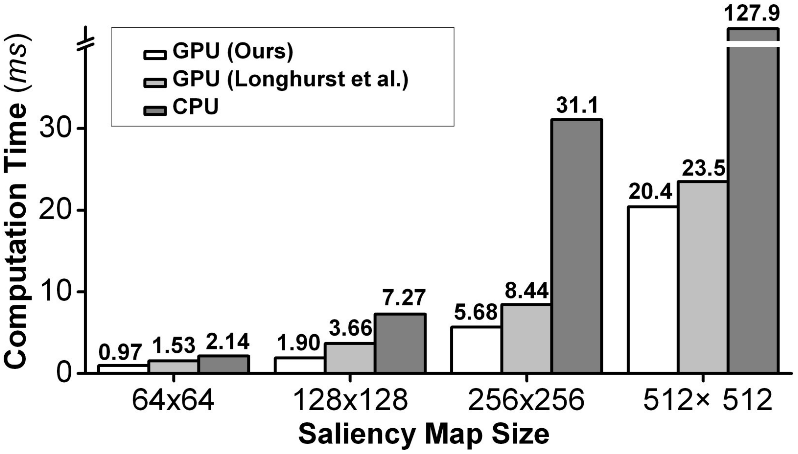


Fig. 3.5 Comparison of measured computation times for generating a saliency map, between the CPU-based and GPU-based (ours and Longhurst et al.'s [60]) implementations.

tween 2.20 times (for the 64×64 image) and 6.27 times (for the 512×512 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 sec.).

3.5 Evaluation of 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.

3.5.1 Methods

A participant's eye movements were recorded using a monocular eye tracking device (Arrington Research Inc.; see Fig. 3.6(a)) with a 60 Hz sampling rate and 640×240 video resolution. A 42-inch LCD display with a resolution of 1,600×900 was used for the presentation of visual scenes (see Fig. 3.6(b)). The participant, wearing the eye tracker, was

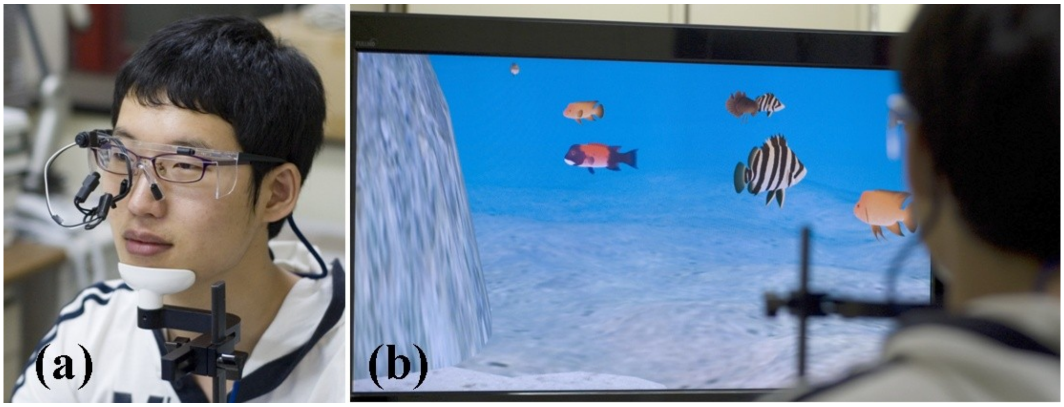


Fig. 3.6 A monocular eye tracker (Arrington Research, Inc.) and a 42-inch LCD display used in the experiment.

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 the upper row of Fig. 3.7), where 30 animated fishes (of six types, with five fishes per type) were swimming. The static environment was a virtual art gallery shown in the lower row of Fig. 3.7.

16 paid subjects (15 male and 1 female) 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 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. 3.8 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 spacebar on the keyboard 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

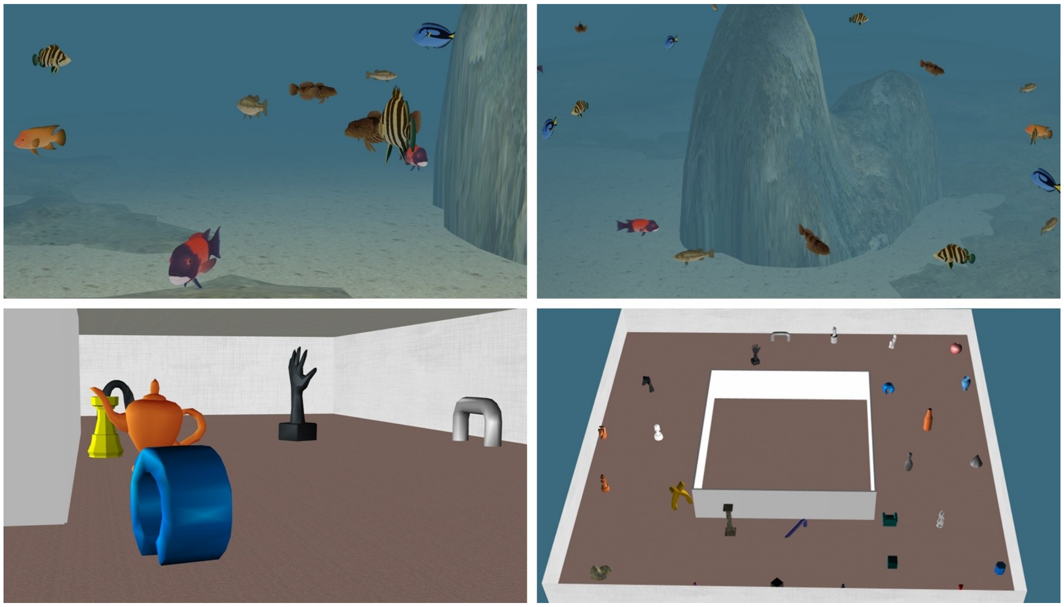


Fig. 3.7 The virtual undersea (dynamic; upper row) and virtual gallery (static; lower row) environments used in the experiment.

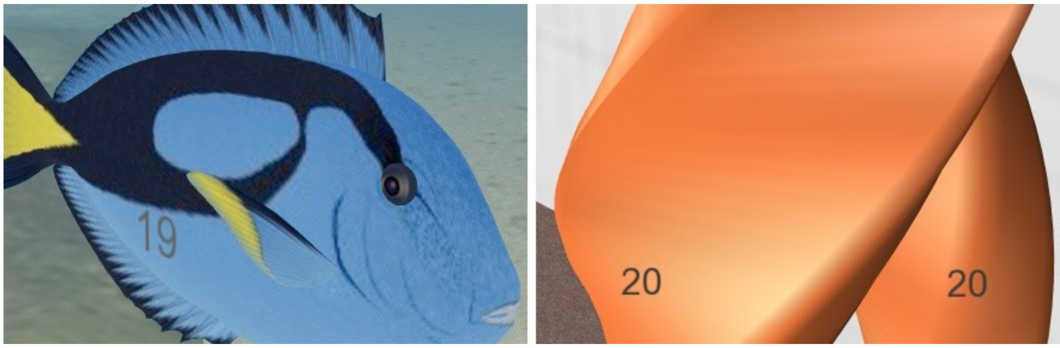


Fig. 3.8 Two examples of number-attached objects. For fishes in the virtual undersea, tags were attached to both sides of the abdomen. For the virtual gallery, they were four tags around the middle section of an object.

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 galley 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 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, oval-fitted centers 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 or her head, and this was facilitated with a chin rest (see Fig. 3.6(a)). 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.

3.5.2 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 attention estimation 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 [17, 75, 4, 89] or the limited precision of the eye tracker.

We also investigated the relative contributions of features to attention estimation. The features were classified into three groups: image features (B: $\{B_l, B_h\}$), extended 3D/object features (E: $\{B_d, B_s, B_m\}$), and top-down context (T: $\{T_s, T_t\}$). To investigate the role of each feature group, attention maps were generated with and without the feature group, resulting in a total of eight (2^3) 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 [93]. To avoid this problem, we simulated the baseline attention map by randomly choosing a pixel on the screen as the most attentive pixel.

3.5.3 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. 3.9, including the standard errors as the error bars. 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.

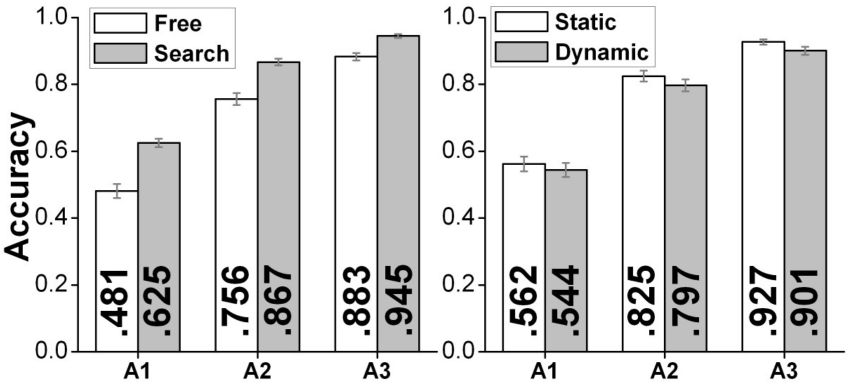


Fig. 3.9 Means and standard errors of the three estimation accuracies (A_1, A_2, A_3) for the task (free navigation versus visual search) and the environment (static versus dynamic) factors.

Table 3.1 Source table of the within-subject 2×2×2 ANOVA, for the effects of task (K) and environment (V).

	A_1		A_2		A_3	
	$F_{1,15}$	p	$F_{1,15}$	p	$F_{1,15}$	p
K	34.75*	<.0001	30.64*	<.0001	22.80*	.0002
V	1.31	.2700	3.36	.0866	7.40*	.0158
$K \times V$	3.59	.0776	6.93*	.0188	22.17*	.0003

Note: $^*p < .05$.

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 3.1 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 effect tests conducted as post-hoc analysis showed that, for free navigation, the mean accuracies in the static environment were significantly higher than those in the

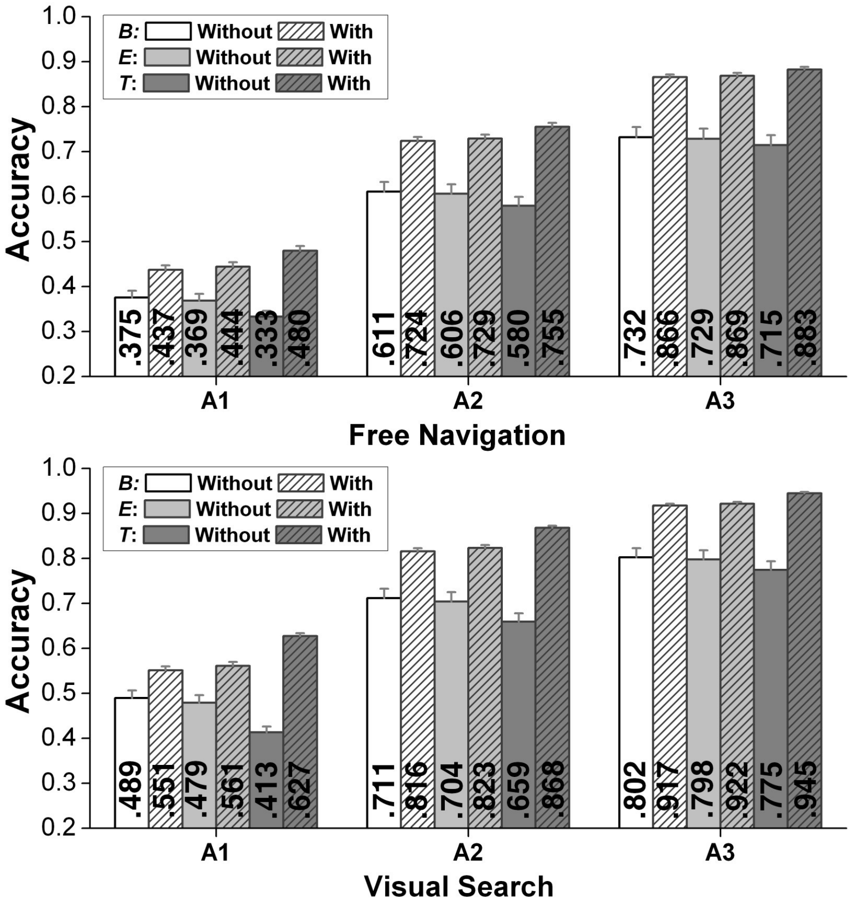


Fig. 3.10 Means and standard errors of the accuracies with and without B, E, and T for free navigation (upper) and visual search task (lower), respectively.

dynamic environment (mean differences = 0.061 and 0.055, p = .0079 and .0004 for A_2 and A_3, respectively), whereas the differences between them were insignificant under the visual search task condition (p = .7957 and .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 A_1, A_2, and A_3 for the main factors, B, E, and T, in Fig. 3.10. As expected, overall accuracies for visual search were

Table 3.2 Means and standard errors of the estimation accuracies for the eight combinations of feature groups (B, E, and T).

B	E	T	A_1	A_2	A_3
0	0	0	.156(.006)	.272(.010)	.359(.012)
1	0	0	.432(.011)	.726(.011)	.866(.008)
0	1	0	.459(.011)	.746(.011)	.880(.008)
1	1	0	.446(.011)	.734(.011)	.874(.008)
0	0	1	.561(.016)	.815(.013)	.915(.008)
1	0	1	.547(.015)	.808(.012)	.918(.008)
0	1	1	.554(.015)	.812(.012)	.914(.007)
1	1	1	.553(.015)	.811(.012)	.914(.007)

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 3.2. 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 A_1) to 0.918 (B and T for A_3). 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 ($p < .0001$) for all accuracies.

3.5.4 Discussion

As discovered in earlier psychological studies [17, 75, 4, 89], human visual attention can be directed at multiple objects (or split foci) 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 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 post-hoc 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 top-down context models, 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 partially estimated 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 3.2) 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%. This is due to the fact that the VE used for the experiment included relatively large objects.

Chapter 4

Real-Time Depth-of-Field Rendering Using Anisotropically Filtered Mipmap Interpolation

This study aims at rendering DOF effects primarily for interactive virtual environments (VEs). The most crucial requirement for employing DOF effects in virtual reality (VR) is to guarantee sufficient and constant rendering performance; at least 60 frames/sec is required for DOF rendering alone, considering time for other operations such as stereoscopy, collision detection, and simulation of other modalities. Among the previous methods, the GPU-based post-filtering approach using downsampled images (called the *pre-blurring* approach) [26, 83, 87, 39, 29] is most plausible for this purpose. However, their quality loss in exchange for rendering speed has been a problem for the use in VR applications.

This chapter presents a DOF rendering technique, based on the post-filtering, that accomplishes both remarkable real-time performance and acceptable image quality appropriate for VR applications. Four representative images rendered by our method are shown in Fig. 4.1. Our method extends the previous pre-blurring approach to one using a generalized *mipmap interpolation* approach. Furthermore, we achieve convincing image quality by significantly reducing three primary visual artifacts often present in the pre-blurring approach: *bilinear magnification* artifact (a magnification artifact occurring due to the bilinear interpolation of

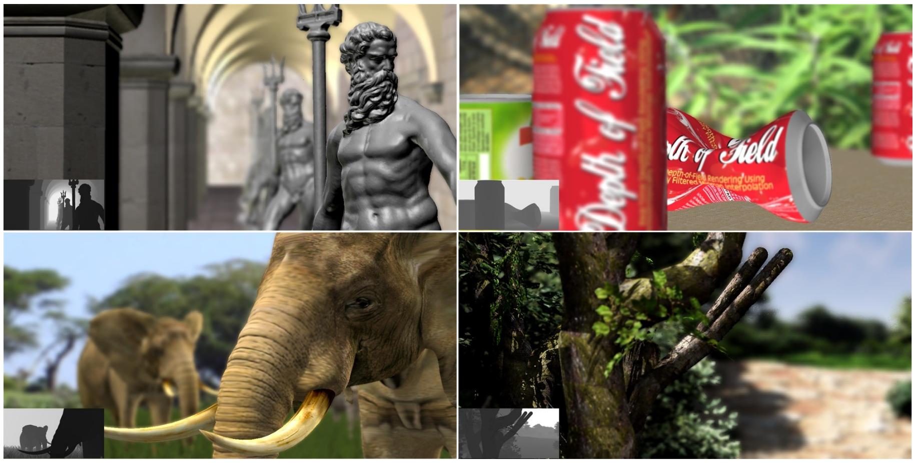


Fig. 4.1 Example images generated with our DOF rendering method and their depth maps (in the bottom-left corner of each image). The three scenes except the “Forest” scene (at the bottom right) were rendered in real time. The “Forest” scene was postprocessed in real time from an offline rendered image. Our method generates realistic DOF effects without any noticeable bilinear magnification artifacts, intensity leakage, and blurring discontinuity.

downsampled images), *intensity leakage* (pixel intensities in a focused area flowing into a blurred area), and *blurring discontinuity* (discontinuity around blurred boundaries in front of a focal plane). Major contributions of this chapter are summarized in what follows.

First, we propose a *nonlinear* mipmap interpolation technique for efficiently generating a color with an arbitrary blurring degree for each pixel in an image. Most of the previous pre-blurring techniques [26, 83, 87, 39, 29] approximate the thin-lens model [81] using linear interpolation of original and downsampled images, while the blurring degree determined by a CoC is nonlinearly related to the depth of a pixel. Our formulation accurately follows the classic thin-lens model, as originally proposed by Potmesil et al. [81].

Second, we propose a filtering method for suppressing the bilinear magnification artifact. Since the pre-blurring approach uses a texture lookup capability in the GPU, blocky magnification artifacts commonly occur when one or two samples are used for a blurred pixel [26, 39, 29]. Some of the previous methods [83, 87] tried to reduce the artifact using Poisson disk sampling, but introduced another artifact called “ghosting” that refers to the overlap of several copies of pixels. Our idea for this problem is to use special filtering where the neighbor pixels for filtering are sampled in *circular* positions at a *one-level lower* mipmap.

Third, we propose an *anisotropic mipmapping* scheme for reducing the intensity leakage. Since most post-filtering methods use spatial filtering (or downsampling) regardless of pixel depths, the blurred image inherently contains intensity leaks from sharp pixels. Although a few previous methods based on heat diffusion [10, 49, 52] attempted to reduce this artifact in the original resolution, additional computational cost made their real-time use very difficult. Inspired by the Perona-Malik pyramid [79, 1], we efficiently preclude the effects of adjacent pixels on the leakage through mipmap construction.

Fourth, we propose an improved and effective method for reducing the blurring discontinuity (or depth discontinuity [26]) at blurred regions in foreground. The discontinuity occurs due to missing directional information from different viewpoints within a lens (also called *partial occlusion* [34]). This is an inherent limitation of all post-filtering methods that use a single viewpoint, but such an anomaly should be alleviated for acceptable visual quality. The previous technique such as blurring of a depth buffer [10] may cause artifacts around the blurred boundaries. Our method blurs the amount of blurring instead of the depth, and ensures the blurred boundaries in foreground look smooth.

Finally, all the methods listed above are completely accelerated using the GPU. Most essential operations including blurring and intensity leakage prevention are performed in reduced resolutions. The computation of anisotropic filter weights is a slight modification of Gaussian weights. Therefore, our method is greatly faster than the previous methods that have provided acceptable image quality, and the performance is comparable to the pre-blurring techniques. For example, our method can produce DOF effects 180 frames/sec at a 1024×768 resolution for 3D models with 287k triangles, indicating sufficient performance for typical VR applications.

The rest of this chapter is organized as follows. Section 4.1 describes our basic framework using the nonlinear mipmap interpolation for real-time DOF rendering. Sections 4.2 and 4.3 extend the basic framework to reduce intensity leakage and blurring discontinuity, respectively. Implementation details are provided in Section 4.4, followed by rendering results and performance statistics in Section 5.4. Section 4.6 discusses the applicability of our work to VR systems and its current limitations.

4.1 DOF Rendering Based on Nonlinear Mipmap Interpolation

This section describes the basic structure of our GPU-based DOF rendering algorithm. Given an image rendered with a pinhole camera model, the depth of a pixel is mapped to a continuous mipmap level, and the final pixel color with the DOF effect is computed by nonlinearly interpolating the pixel colors from discrete-level mipmap images. This basic framework is further extended in Sections 4.2 and 4.3 for reducing intensity leakage and blurring discontinuity, respectively.

4.1.1 Construction of Mipmap Pyramid

We first render a scene to two textures (one for color and the other for depth) whose dimension is the same as a frame buffer. A 16-bit floating-point texture is used to maintain the precision of depth computation. Our framework uses the magnification of mipmap images downsampled from the color texture to simulate spatial filtering. Since a hardware-generated mipmap exhibits awkward aliasing in the enlarged images, we employ Gaussian filtering during downsampling. The depth texture is used for computing a mipmap level required for nonlinear mipmap interpolation.

Let $I(l)$ be a mipmap image at level l and $\mathcal{G}$ be a Gaussian convolution operator with the kernel size of 3×3. Beginning from an input image (l=0), the next level (coarser) mipmap is rendered onto a quarter-sized image by

$$I_{\mathbf{p}}(l) = \mathcal{G} * I_{\mathbf{p}}(l-1) = w_G \sum_{\mathbf{q}\in\Omega} G(\mathbf{q}-\mathbf{p}) I_{\mathbf{q}}(l-1), \tag{4.1}$$

where $\mathbf{p}$ is a center pixel position, Ω is a set of the center and eight neighbor pixels, $G(\mathbf{x})$ is the typical Gaussian weight at an offset $\mathbf{x}$. w_G is the reciprocal of the sum of the Gaussian weights.

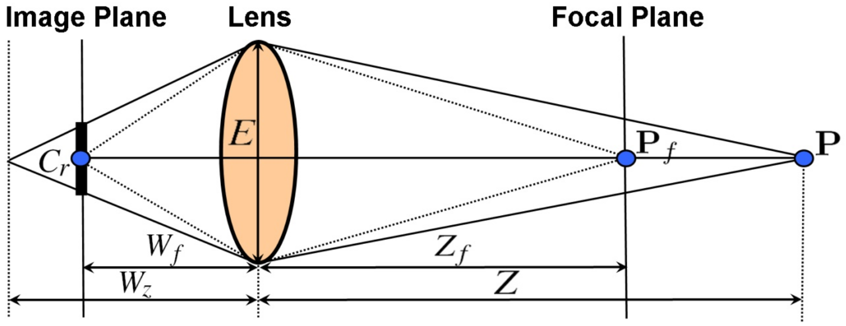


Fig. 4.2 The thin-lens model [81].

4.1.2 Computing CoC from Depth

Fetching a color from the constructed mipmap for each pixel begins with the computation of CoC size from its depth using the thin-lens model shown in Fig. 4.2. Here, the image distance, V, of an object located at $\mathbf{P}$ and the image distance, V_f, of a focal position at $\mathbf{P}_f$ are related to the configurations of the lens and the object by

$$V = \frac{Fd}{d-F} \quad \text{and} \quad V_f = \frac{Fd_f}{d_f - F}, \tag{4.2}$$

where F is the focal length of a lens (typically 16.7 mm in an adult for an object at infinity [57]). d and d_f are the depths at $\mathbf{P}$ and $\mathbf{P}_f$, respectively. Then, for an effective lens size E, the diameter of the CoC on the image plane (or the human retina), R, can be computed as

$$R = |V - V_f|\frac{E}{V} = \left(\frac{EF}{d_f - F}\right)\frac{|d-d_f|}{d}. \tag{4.3}$$

R can be further projected onto the diameter of the CoC on the screen in the pixel-space, C:

$$C = DPI\frac{d_s}{d_r}R, \tag{4.4}$$

where DPI is the number of pixels per unit length (in our implementation, dots per mm), and d_s and d_r denote the distances from the lens to the screen and the image plane, respectively. Typically, d_r=24 mm for an adult [84] is used in our computation.

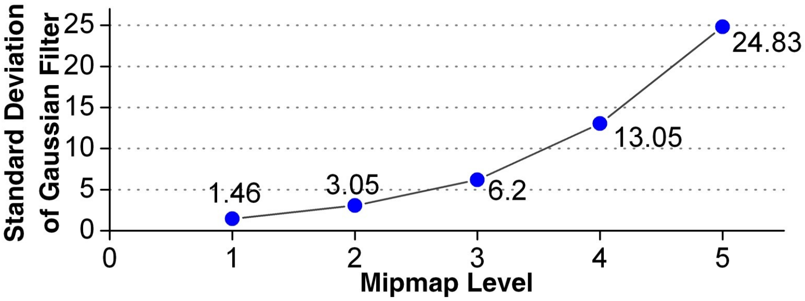


Fig. 4.3 A mapping from mipmap levels to the standard deviations of a Gaussian filter that produces the most similar images in terms of RMSE.

4.1.3 Relating CoC to Mipmap Level

The pixel-space CoC diameter, C, represents the degree of blurring obtained by Gaussian filtering at the original resolution. In our approach, the Gaussian filter is approximated using magnification from a mipmap level to its original resolution. Since the blurring degree is determined by the intensity distribution function in a CoC [81, 18], we should choose a specific lens, such that its optical property matches the Gaussian intensity distribution with a standard deviation, σ. In our implementation, a lens with $\sigma = C/2$ is chosen, and σ is called the *degree of blurring* (DoB).

A quantitative relationship between σ and a corresponding mipmap level, m, can be found by finding a best match in terms of pixel-wise root-mean-square error (RMSE) between a blurred image using the Gaussian filter and an image magnified from the mipmap. Average results empirically obtained with the “Lena” and “Baboon” images are provided in Fig. 4.3. The relation between m and σ can be approximated by

$$\sigma = \frac{3}{2} \cdot 2^{m-1}. \tag{4.5}$$

Using this rule, we can define a continuous mipmap level index, m, related to σ, as

$$m = \left\{ \begin{array}{ll} 0 & \text{if} k_\sigma \sigma < 0.5 \\ 1+\log_2(k_\sigma \sigma) & \text{otherwise} \end{array} \right. , \tag{4.6}$$

where k_σ (= $\frac{2}{3}$ in our implementation, from (4.5)) is a scaling constant. The values of σ such that $k_\sigma \sigma < 0.5$ represent that the corresponding filtering has no blurring effect.

4.1.4 Nonlinear Mipmap Interpolation with Circular Filtering

An adequately blurred pixel color can be obtained by magnifying a mipmap image at m to the original resolution. A major difficulty in this step stems from the fact that, although m is defined in a continuous range, a mipmap has discrete levels. Thus, an interpolation using two integer mipmap levels, $l = \lfloor m \rfloor$ and $l+1$, is required.

Once l and m are determined for a pixel $\mathbf{p}$, a straightforward interpolation can be

$$I_{\mathbf{p}}(m) = I_{\mathbf{p}}(l)^{1-\beta} I_{\mathbf{p}}(l+1)^{\beta}, \text{where } \beta = m - l. \tag{4.7}$$

However, when this interpolation is implemented using the built-in bilinear magnification, it may exhibit some blocky artifacts that can cause temporal flickering in dynamic scenes. This magnification can be improved using cubic interpolation [11], but with heavy computation.

Our strategy is to blend neighbors at the one-lower mipmap level, m–1:

$$I_{\mathbf{p}}(m) = w_G \sum_{\mathbf{q} \in \Omega} G(\mathbf{q} - \mathbf{p}) I_{\mathbf{q}}(l-1)^{1-\beta} I_{\mathbf{q}}(l)^{\beta}. \tag{4.8}$$

This equation uses *finer* mipmap images that contain more image details, and thus, results in magnification quality much better than that of cubic interpolation using the source pixels at level m. Moreover, this technique requires only eight additional texture lookup operations per pixel compared to the 16 operations of cubic interpolation (one for weights and 15 for neighbor pixel colors).

Another technique we use for quality improvement is to change the sampling positions to follow a unit circle as shown in Fig. 4.4b. In general, the bilinear interpolation artifacts are worst at pixels corresponding to edges in a lower resolution image. When such a pixel is sampled from a non-edge position, the artifact can be suppressed. For example, a pixel at the upper right is sampled from $(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})$ instead of $(1,1)$ (see Fig. 4.4). We find that this simple technique greatly improves magnification quality. Fig. 4.5 shows a comparison between images magnified using the bilinear magnification and our method.

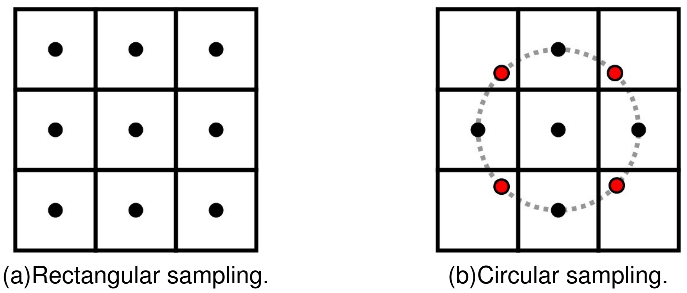

(a)Rectangular sampling. (b)Circular sampling.

Fig. 4.4 A comparison between (a) rectangular sampling and (b) our circular sampling for Gaussian filtering. In the circular sampling, diagonal samples (red points) are located more closely toward the center.

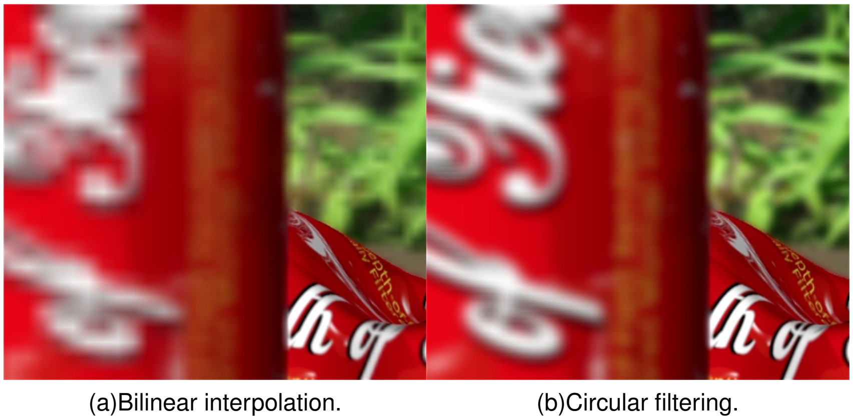
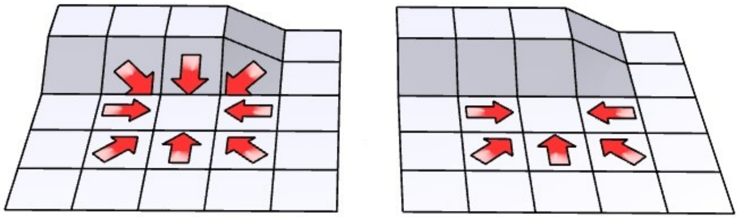

(a)Bilinear interpolation. (b)Circular filtering.

Fig. 4.5 A comparison of DOF effects with (a) bilinear magnification and (b) our magnification method with finer mipmap images and the circular blending. The bilinear magnification yields severe jagged artifacts, while ours almost removes the artifacts and well maintains the object shapes.

4.2 Anisotropic Mipmap Construction for Intensity Leakage Reduction

In this section, we extend the basic framework (described in Section 4.1) to further reduce intensity leakage using an anisotropic Gaussian filter. Since our anisotropic filter, unlike the isotropic mipmapping, uses DoB (σ) and depth values at each mipmap level, we also build isotropic mipmaps for depth and DoB, and denote them by Z and B, respectively.

In principle, intensity leakage occurs when focused pixels are blended during Gaussian



(a)Isotropic Gaussian filter. (b)Anisotropic Gaussian filter.

Fig. 4.6 An illustration of the two 3×3 Gaussian filters used during mipmapping. The elevated pixels represent sharp pixels (lower σ than other pixels), and the arrows indicate the direction of intensity flow.

filtering for pixels behind the focal plane (i.e., background), whereas the focused pixels have small CoCs and thus hardly affect their neighborhood. Hence, if we preclude the effects of the focused pixels when building the mipmap of color, the leakage can be effectively removed. Fig. 4.6 illustrates how pixel intensities flow during building a mipmap with 3×3 isotropic and anisotropic Gaussian kernels. In Fig. 4.6b, if a neighbor pixel has a DoB lower than its center pixel (higher sharpness; $\sigma_{\mathbf{q}} < \sigma_{\mathbf{p}}$), the pixel is not blended, because it has a smaller CoC than the center pixel (the upper three pixels). On the other hand, if a neighbor pixel has a DoB larger or equal to that of the center pixel (less sharp; $\sigma_{\mathbf{q}} \geq \sigma_{\mathbf{p}}$), the pixel is blended for proper blurring (the other five pixels).

This behavior can be implemented by simply replacing the isotropic Gaussian operator $\mathcal{G}$ in (4.8) with an anisotropic Gaussian operator $\mathcal{H}$ defined as

$$H(\mathbf{p},\mathbf{q},l) = \begin{cases} 0 & \text{if } \sigma_{\mathbf{p}}(l) > \sigma_{\mathbf{q}}(l) + \delta \text{and } Z_{\mathbf{p}}(l) > Z_f \\ G(\mathbf{q}-\mathbf{p}) & \text{otherwise} \end{cases}, \tag{4.9}$$

where $\sigma_{\mathbf{q}}(l)$ and $\sigma_{\mathbf{p}}(l)$ denote the DoBs at mipmap level l, and $Z_{\mathbf{p}}(l)$ the depth at l. δ is a slight offset for allowing neighbor pixels with similar DoB values to be included in blurring. The second condition ($Z_{\mathbf{p}}(l) > Z_f$) means that the leakage is prevented only for background pixels. Fig. 4.7 shows comparative results generated by the isotropic and anisotropic mipmapping.

The use of the anisotropic Gaussian filter is better than the previous pre-blurring approach [26, 83, 87, 39, 29] in terms of resulting image quality and than the anisotropic

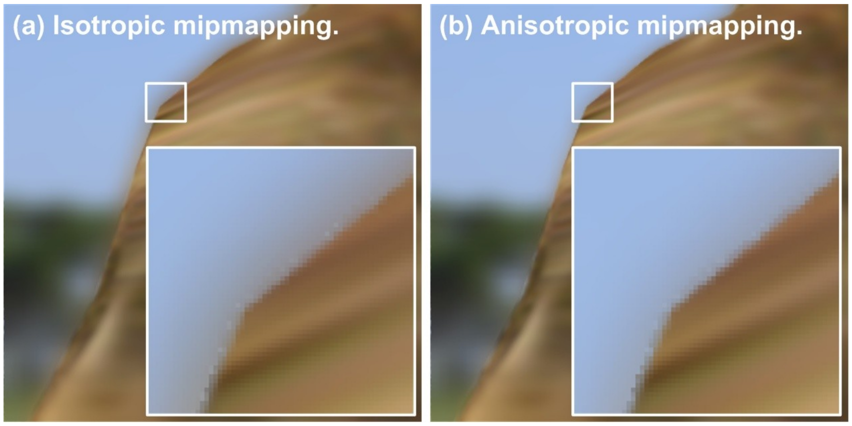


Fig. 4.7 A comparison of the results generated by the isotropic and anisotropic Gaussian mipmapping. The intensity leakage around the sharp area (leaked brown colors in the left image) was greatly reduced in the right image.

diffusion approach [10, 49, 52] in terms of rendering performance. Since the downsampled images already contain intensity leakage, remedies in the magnification step [87] cannot be very efficient nor accurate. Instead, we have chosen to *preclude* the leakage during downsampling. The anisotropic diffusion requires a great deal of iteration, even for a moderately blurred scene. In contrast, our method performs leakage prevention in lower resolution, and the computational cost is only dependent on the number of mipmap levels. As a consequence, our method guarantees stable performance suitable for real-time applications, with significantly reduced intensity leakage.

The anisotropic Gaussian filter can also produce a minor artifact. The filter is a kind of simplified bilateral filter [3, 92] and tends to preserve edges around focused and background areas, although background edges should be filtered out for proper blurring effects. We, however, note that the preserved background edges cannot be easily perceived when magnified to the original resolution, due to the smoothing effect of the magnification itself.

4.3 Smoothing Foreground Boundaries

In this section, we explain a method for smoothing discontinuous foreground boundaries. Since the majority of post-filtering techniques use a single color/depth image, partially visible areas behind foreground objects cannot be accurately rendered. This limitation may cause objectionable artifacts where blurred boundaries around the foreground look discontinuous. We alleviate the artifact, without additional visibility information, by improving the previous depth buffer blurring method [10] and merging it to our mipmap framework. Note that reading a pixel from a specific mipmap level corresponds to blurring of the pixel.

In our framework, a partial occlusion effect can be approximated by fetching blurred colors for pixels around the foreground and its boundaries. In (4.6), a mipmap interpolation level, m, was determined from a discontinuous DoB value, $\sigma(0)$ (see Fig. 4.8a). Hence, if we fetch σ from a higher mipmap level, n, instead of $\sigma(0)$, the foreground boundaries can be smoothed (see Fig. 4.8b). In addition, n should be higher for only around foreground objects to maintain non-occluded areas sharply.

Observing that a focused pixel can be occluded by nearer foreground neighbor pixels, we store the highest DoB value among the pixels. We call this value the *degree of foreground neighbors' blur* (DoN; ρ), and build the mipmap of DoN, N, as

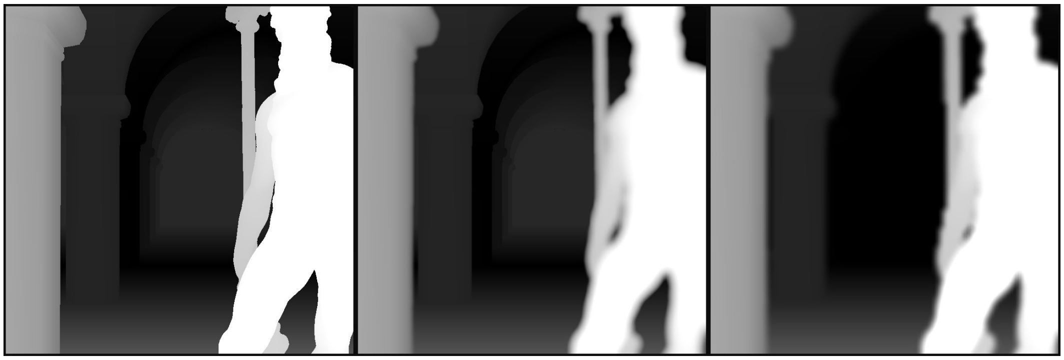
$$\rho_{\mathbf{p}}(l) = \begin{cases} \max\limits_{\mathbf{q}\in\Omega'} \sigma_{\mathbf{q}}(l-1) & \text{if } \Omega' \neq \text{null} \\ 0 & \text{otherwise} \end{cases}, \tag{4.10}$$

where Ω' denotes a subset of Ω containing foreground neighbor pixels ($Z_{\mathbf{q}} < Z_f$) nearer than the center pixel ($Z_{\mathbf{q}} < Z_{\mathbf{p}}$). If a DoN value is taken and magnified (using the circular filtering) at the highest mipmap level, M, this value (i.e., $\rho(M)$; see Fig. 4.8c) can be used for finding the foreground objects and their boundaries that may occlude non-foreground areas.

These conditions are satisfied by defining n as

$$n = \begin{cases} 0 & \text{if } k_\sigma \rho(M) < 0.5 \\ 1+\log_2(k_\sigma \rho(M)) & \text{otherwise} \end{cases}. \tag{4.11}$$

This is equivalent to (4.6) but applied to the mipmap of DoB instead of color. As a result, n for a foreground pixel is determined from a higher (coarser) mipmap level ($k_\sigma \rho(M) > 0.5$;

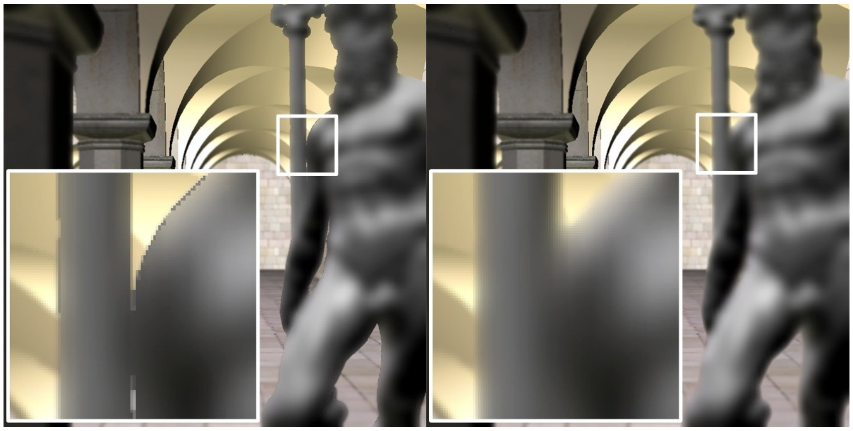


(a)$\sigma(0)$ (b)$\sigma(n)$ (c)$\rho(M)$

Fig. 4.8 DoB and DoN textures: (a) an original DoB image, (b) a DoB image foreground-smoothed using $\rho(M)$, and (c) a DoN image read from the highest mipmap level, M. Note that the background pixels in $\rho(M)$ are black (i.e., $\rho(M)$=0). The images were scaled for illustration.

non-black pixels in Fig. 4.8c) of B, and n is smoothly changed from the foreground to the boundaries. On the other hand, n for a background or focused pixel is determined from the level 0 ($k_\sigma \rho(M) \leq 0.5$; black pixels in Fig. 4.8c), which allows non-foreground DoBs to be maintained sharply. As a consequence, a DoB image is smoothed for only foreground and its boundaries, and the discontinuity can be practically removed. Fig. 4.9 highlights the difference between images rendered using discontinuous and smoothed DoBs.

Compared to the blurring of a depth buffer [10], our smoothing technique is a great improvement in the following aspects. First, our method does not cancel DoBs where the foreground and background meet, owing to the blurring of DoB instead of depth. Second, our method using the circular filtering shows no bilinear artifacts in the blurred DoBs. Finally, our method can smoothly blend the occluded focused area with the non-occluded focused area, whereas the depth blurring method cannot blend the two areas easily (no details presented in [10]).



(a)Discontinuous foreground. (b)Smoothed foreground.

Fig. 4.9 A comparison of the results generated with (a) discontinuous DoBs and (b) smoothed DoBs using our method. The blurring discontinuities are significantly reduced in the right image.

4.4 Implementation Details

The DOF rendering algorithm presented has been implemented using the OpenGL and the OpenGL Shading Language [50]. The computational procedure consists of five steps as illustrated in Fig. 5.2 with the circled numbers representing each step.

In step 1, the graphics engine renders color, depth, and DoB to two floating-point textures, $I(0)$ and $ZBN(0)$ (DoN is not defined in this step). The three channels (usually for "red-green-blue") are used for Z, B, and N. In order to avoid redundant model rendering, $I(0)$ and $ZBN(0)$ are simultaneously rendered using multiple-render-target (MRT) capability in the GPU, which involves the use of multiple textures as output targets.

In step 2, ZBN is built in parallel using $ZBN(0)$ as a source. Z and B are downsampled using the isotropic Gaussian filtering, and N is down-sampled using a maximum DoB value among foreground neighbor pixels. The rendering of sub-maps begins from the level one (i.e., quarter-sized map), and continues to the level corresponding to the maximum DoB, σ_{max}.

In step 3, I is built using the anisotropic Gaussian filtering. During the downsampling,

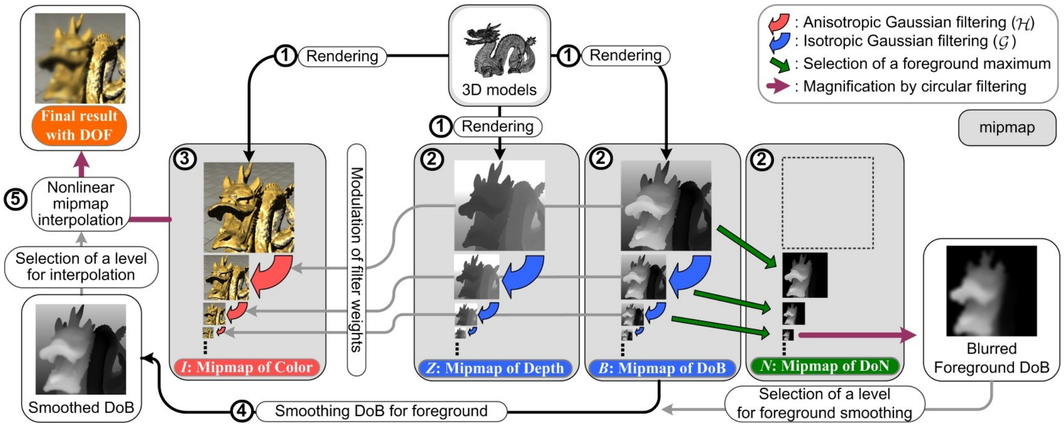

Final result with DOF
① Rendering
3D models
① Rendering
① Rendering
: Anisotropic Gaussian filtering ($\mathcal{H}$)
: Isotropic Gaussian filtering ($\mathcal{G}$)
: Selection of a foreground maximum
: Magnification by circular filtering
mipmap
⑤ Nonlinear mipmap interpolation
Selection of a level for interpolation
Smoothed DoB
③
I: Mipmap of Color
Modulation of filter weights
②
Z: Mipmap of Depth
②
B: Mipmap of DoB
②
N: Mipmap of DoN
Blurred Foreground DoB
④ Smoothing DoB for foreground
Selection of a level for foreground smoothing

Fig. 4.10 The overall framework of our DOF rendering applied at every rendering frame.

the depth and the DoB values read from ZBN are used for computing the anisotropic filter weights. In practice, steps 2 and 3 can be simultaneously executed in a single shader using MRT capability to avoid redundant rendering and texture lookup operations for depth and DoB.

In step 4, the DoB image, $B(0)$, is smoothed using the mipmap of DoN, N. We first read and magnify the DoN value, $\rho(M)$ (i.e., $N(M)$), from the highest mipmap level with the circular filtering (using 5-point or 9-point samples). Next, we read the DoB value, $\sigma(n)$, from the mipmap level, n, that corresponds to $\rho(M)$. This smoothed DoB, $\sigma(n)$, is used to determine the interpolation level, m, for color.

In step 5, a result image with a DOF effect is produced by interpolating the two pixel values read from the color mipmap. To remove the bilinear artifacts we sample 16 (2 levels $\times$ 8 offsets) additional neighbors using the circular offsets. The two texture lookup operations for reading pixels in the adjacent mipmap levels can be performed in one operation using the mipmap biasing [50] that directly reads a sub-image at a non-integer mipmap level. Thus, 18 texture lookup operations are reduced to 9 operations. We tested and confirmed that the difference between the results of the mipmap interpolation and the mipmap biasing is marginal.

4.5 Results

In this section, we report the image quality and rendering performance of the proposed DOF rendering framework and compare them to those of recent GPU-based post-filtering methods.

4.5.1 Rendering Quality

We have compared our implemented framework to recent real-time post-filtering techniques: Kraus et al. [53], Zhou et al. [103], and Scheuermann [87], which are based

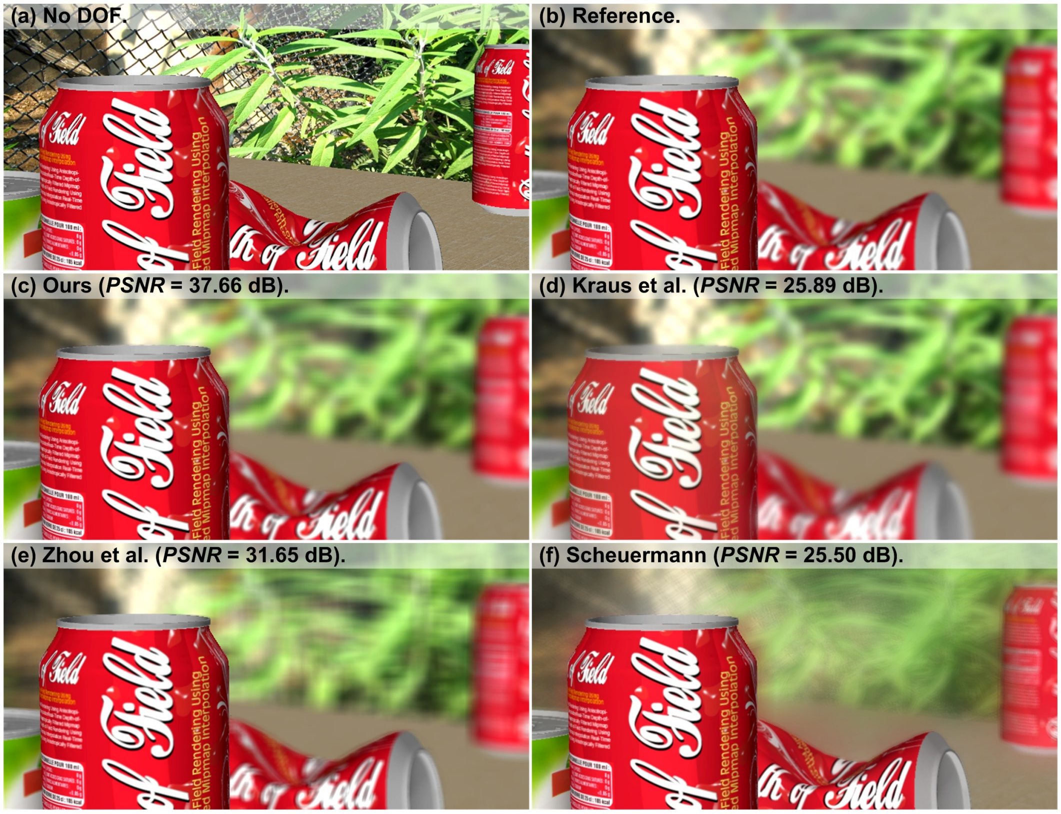

(a) No DOF.
(b) Reference.
(c) Ours (*PSNR* = 37.66 dB).
(d) Kraus et al. (*PSNR* = 25.89 dB).
(e) Zhou et al. (*PSNR* = 31.65 dB).
(f) Scheuermann (*PSNR* = 25.50 dB).

Fig. 4.11 A comparison of background-blurred images rendered by our method and three recent GPU-based methods [87, 103, 53]. The nearest can is focused. The reference image was rendered by the accumulation buffer method [43], and used for computing $PSNR$s.

on the pyramidal image processing (similar to our mipmap approach but using multiple layers), the separable Gaussian filtering, and the pre-blurring, respectively. The accumulation buffer method [43] was also implemented for the reference of image quality. We did our best to balance minor differences among the methods for fair comparisons; for example, in Kraus' method we used eight layers for producing similar blurring degrees. In addition to subjective comparisons, visual quality was assessed in terms of peak signal-to-noise ratio ($PSNR$) that is a common objective quality metric. The $PSNR$ was computed as $10\ \log_{10}(255^2/MSE)$ where MSE represents the mean squared error for the reference image rendered by the accumulation buffer method.

We first report the image quality of background blurring, primarily to illustrate intensity leakage. Fig. 4.11 shows the result images for the "Cans" scene focused on the nearest can. The intensity leakage was observed in the result by Scheuermann's method only (red colors are spread over the outer boundary of the focused can; see Fig. 4.11f). Regarding blurring quality, our method generated a natural image without objectionable artifacts including the bilinear magnification artifacts. Other methods, however, revealed some artifacts specific to each. In that of Kraus et al., overall colors seem slightly brighter than the others (see Fig. 4.11d), resulted from the repeated inclusion of identical pixels into multiple layers. That of Zhou et al. appeared to be irregularly blurred and distorted (see Fig. 4.11e), caused by the violation of the separability of the Gaussian filter—their filter is not separable. Scheuermann's failed to achieve accurate blurring degree; their image seemed like a mix of the original and much blurred images rather than an adequately blurred one (see Fig. 4.11f). This is because their method used no intermediate blurring information. The computed $PSNR$s also confirmed the quantitative differences. Our method created an image most similar to the reference ($PSNR$=37.66 dB), whereas the others showed relatively lower $PSNR$s (25.89 dB, 31.65 dB, and 25.50 dB, respectively).

Fig. 4.12 illustrates the quality of foreground-blurred images. The "Sponza Atrium" scene was used for comparison, with a focus on the "Neptune" statue. It can be seen that all methods including ours failed to achieve correct partial occlusion effects (the reference is shown in Fig. 4.12b); particularly, sharp and semitransparent focused areas were not

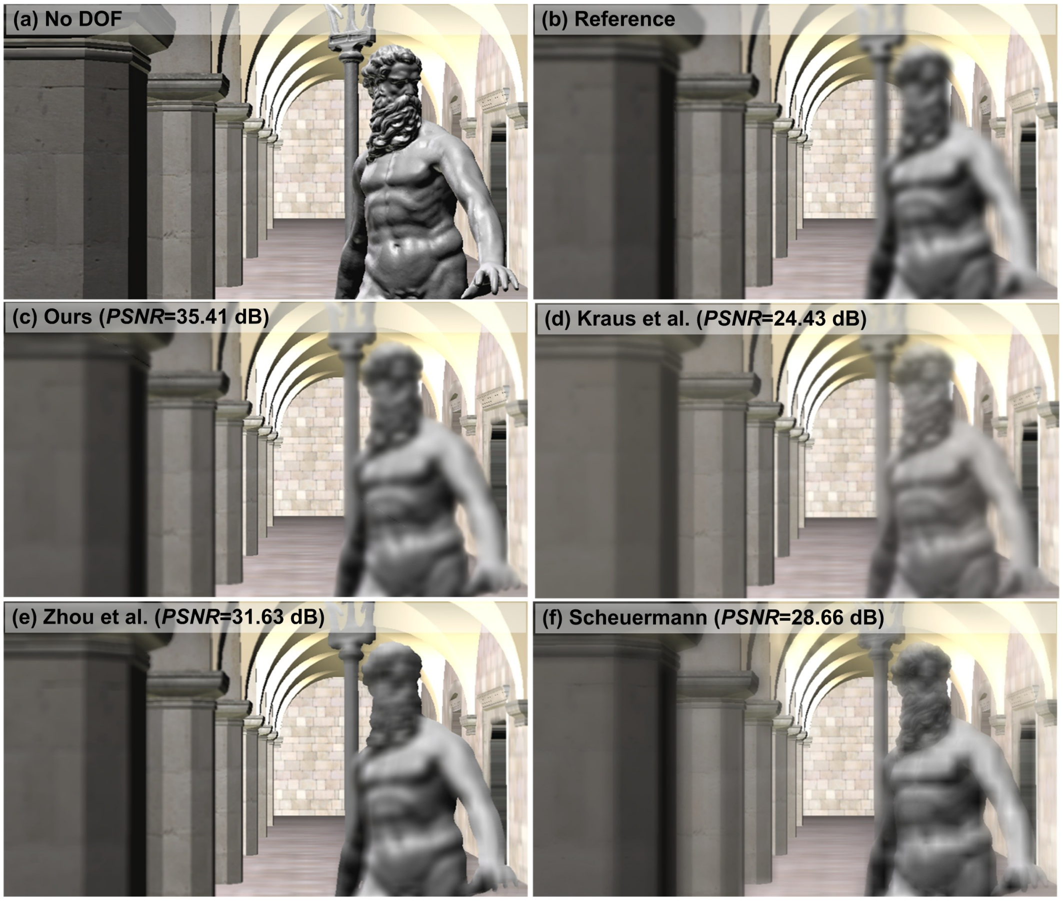

(a) No DOF
(b) Reference
(c) Ours (*PSNR*=35.41 dB)
(d) Kraus et al. (*PSNR*=24.43 dB)
(e) Zhou et al. (*PSNR*=31.63 dB)
(f) Scheuermann (*PSNR*=28.66 dB)

Fig. 4.12 A comparison of foreground-blurred images rendered by the same methods in Fig. 4.11. The farthest wall is focused.

handled correctly. For the blurring discontinuity, the methods of ours and Kraus et al. could maintain smooth boundaries around the foreground objects, while those of Zhou et al. and Scheuermann failed to do so. Moreover, Scheuermann's showed clear bilinear magnification artifacts. Although the method of Kraus et al. has an advantage for partial occlusion effects due to the boundary extrapolation used, it resulted in distorted colors (their image was also brighter than the others) that can degrade the image quality. These quality differences were well reflected into $PSNR$s, where our method showed better image quality (35.41 dB) than the others (24.43 dB, 31.63 dB, and 28.66 dB, respectively).

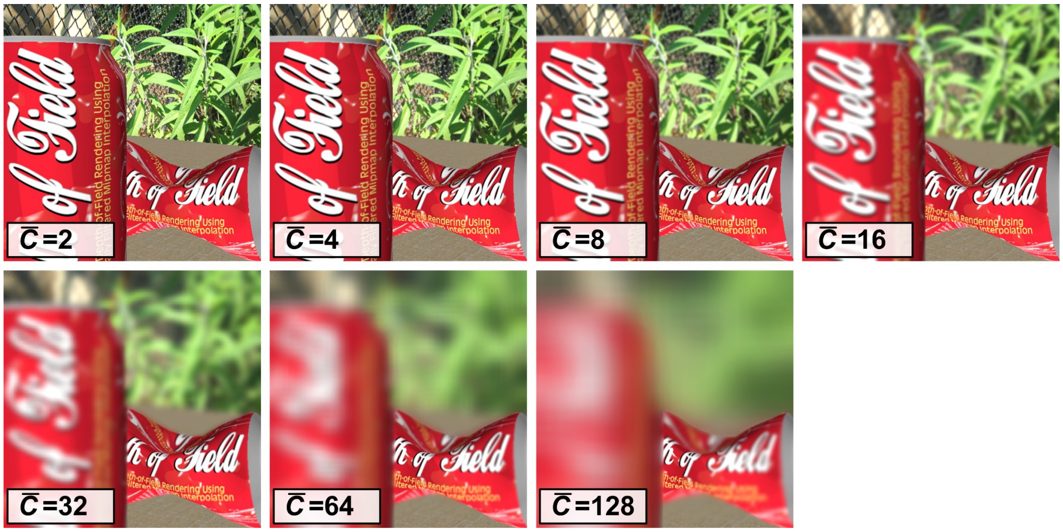


Fig. 4.13 The evolution of blurred appearance. When the mean CoC diameter ($\bar{C}$) is equal to or less than 16 pixels, the artifacts do not stand out.

Our framework is not perfect with regard to partial occlusion. In principle, the focused area partially occluded by foreground objects should be sharply and semitransparently represented (as shown in Fig. 4.12b). Our method, however, may fail especially for high blurring degree. An example is given in Fig. 4.13 where unnatural partial occlusion rendering appears for the mean CoC diameter ($\bar{C}$) larger than 16 pixels. This tendency was common to many cases we tested. This limitation results from the use of a single color/depth image as an input [6]. The mean CoC diameter less than 16 pixels is usually sufficient for simulating DOF effects in a human eye, but can be insufficient for simulating a camera system with a large aperture.

The last quality issue is temporal flickering that may occur when downsampled images for blurring are used in dynamic environments. We have tested each method for interactive navigation. The method of Zhou et al. showed no flickering. In our method, flickering was observed only for much blurred scenes (e.g., the mean CoC diameter larger than 32 pixels), since more than eight neighbor pixels seem to be necessary in our circular filtering for such cases. In contrast, Scheuermann's method exhibited severe flickering due to the bilinear magnification. The same is applied to Kraus' method especially around the boundaries of focused areas, due to the use of coarser images for extrapolation. In summary, the methods of Zhou and ours are adequate for dynamic VEs with respect to temporal flickering.

To summarize, our method simulates realistic DOF effects in an image without noticeable artifacts including bilinear magnification artifact and intensity leakage. Despite missing directional information, image quality on foreground blurring is acceptable for moderately blurred scenes. Furthermore, in such a blurring range, our method does not exhibit temporal flickering in dynamic environments.

4.5.2 Rendering Performance

We measured the rendering performance of the four methods in terms of frame rate. The test was performed on an Intel 2.67 GHz Core2Duo machine with an nVidia GeForce 8800GTX at 1024×768 display resolution. The accumulation buffer method was excluded, since it requires at least 1 sec per image. Four scenes, "Forest", "Cans", "Elephants", and "Sponza Atrium", shown in Fig. 4.1 were chosen for various scene complexity. The numbers of triangles were 2 (a quad for the rendered image), 287,695, 456,114, and 2,061,886, respectively. Focal parameters were controlled so that the mean CoC size, $\bar{C}$, changed from 4 to 128 pixels.

Fig. 4.14 shows the benchmark results. The method of Scheuermann was the highest due to its simplicity. Our method ranked the second, with frame rates higher than 90 frames/sec even for the most complex scene. The performance was also independent from the size of CoC. The method of Kraus et al. was the slowest with the best frame rate lower than 30 frames/sec, indicating that it cannot be used for real-time VR applications. The performance of Zhou et al. had significant correlation with the CoC size. Its frame rate was the highest for small CoCs but rapidly dropped to be slower than our method with increasing CoC. To sum up, our method demonstrated a balanced performance between the speed and the independence from the degree of blur, proving its adequacy to VR applications.

Finally, the results of formal analysis on the computational costs of the four methods with respect to the number of texture lookup operations (the most relevant bottleneck in a

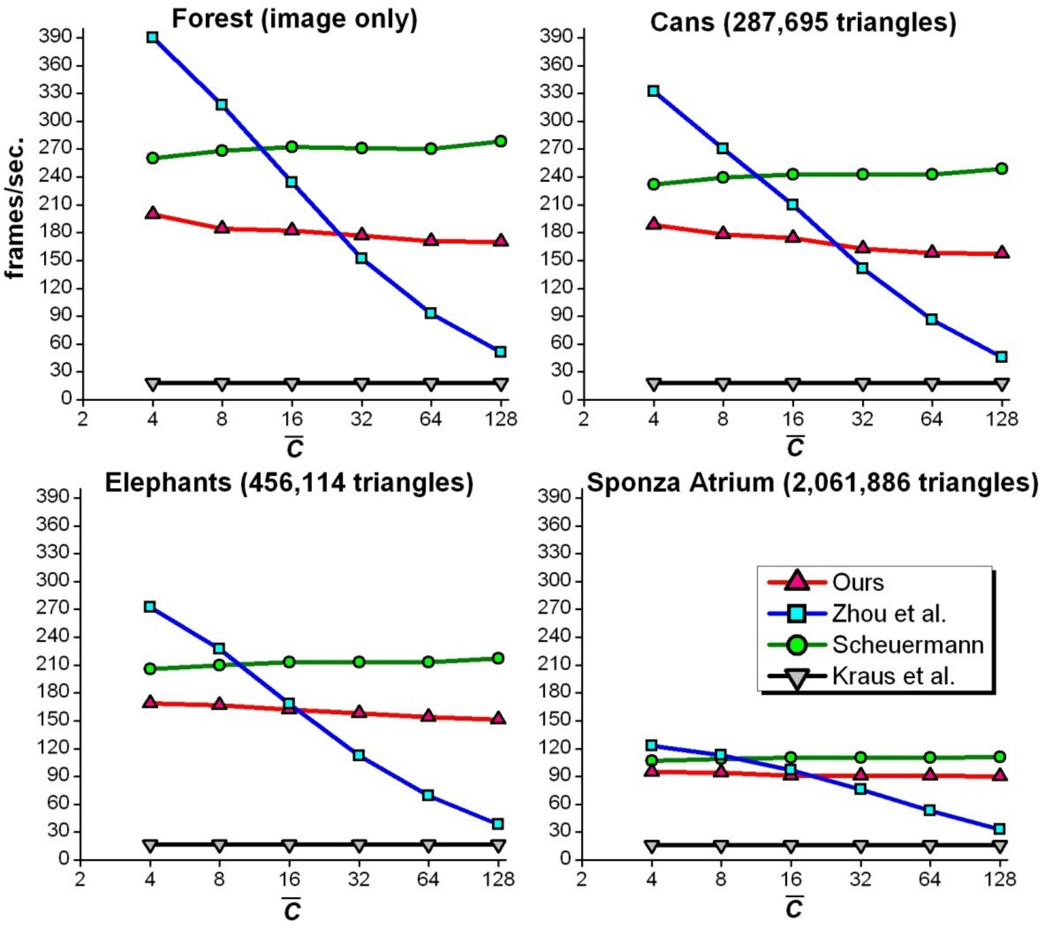


Fig. 4.14 Performance comparison in terms of frame rate for the four methods: ours, Zhou et al. [103], Scheuermann [87], and Kraus et al. [53].

GPU program [10, 103]) are summarized in Table 4.1. A scene where a maximum CoC diameter is 16 pixels at 1024×1024 display resolution is used for the comparison. Operations required for model rendering are excluded. All methods except ours used a RGBA color/depth texture where the depth is stored in the alpha channel, while ours used a ZBN mipmap to read a depth. The table shows that our method requires only 20.98 units (1 unit = 1,048,576 = 1024×1024 operations), which is much smaller than those of the other methods (24.03, 27.00, and 173.02 units, respectively). This is because in our method most opera-

Table 4.1 Examples for the number of texture lookup operations required in GPU programs for the four methods. The largest CoC is 16 pixels ($\sigma_{max} = 8$) at a 1024×1024 display resolution. 1,048,576 ($= 1024 \times 1024$) operations are considered one unit.

Method	Total units	Intermediate operations	Units required to an operation
Ours	20.98	Building the mipmap of depth, DoB, and DoN	$9\times(1/2^2+1/4^2+1/8^2+1/16^2)$
		Building the mipmap of color	$9\times(1/2^2+1/4^2+1/8^2+1/16^2)$
		Smoothing foreground DoB with 5-point circular filtering	6
		Mipmap interpolation (biasing) with 9-point circular filtering	9
[87]	24.04	Downsampling and Gaussian filtering of color to $1/16^2$ size	$9\times(1/16^2)$
		Reading the original color/depth during blending	12
		Reading the downsampled color during blending	12
[103]	27.00	Reading the original color/depth during horizontal filtering	$4\times2+1$
		Reading the blurred color and the original depth during vertical filtering	$2\times(4\times2+1)$
[53]	173.02	Culling of foreground pixels	8
		Analyses for disocclusion and pyramidal blurring	$8\times20\times(1/2^2+1/4^2+1/8^2+1/16^2)$
		Syntheses for disocclusion and pyramidal blurring	$8\times9\times(1+1/2^2+1/4^2+1/8^2)$
		Matting of the disoccluded sub-images	8
		Blending of the blurred sub-images	8

Note. For Scheuermann's, 12-point sampling was used in the final blending step. For that of Zhou et al., the mean CoC radius was assumed to be 4 pixels, requiring 9 ($=4\times2+1$) iterations. Kraus' method used 8 layers. Implementation details for each are found in [87, 103, 53].

tions except the final blending are carried out in coarser resolutions. Our method also has additional overhead for building and copying mipmaps by rendering multiple view-ports, which results in the final performance shown in Fig. 4.14.

4.6 Discussion

We have shown that our DOF rendering framework satisfies both acceptable image quality and sufficient rendering performance that are appropriate to our target, VR applications. Our method yields accurate results that cannot be easily distinguished from those of multipass rendering, except for scenes with a high blurring range (e.g., $\bar{C} > 16$; see Fig. 4.13

again). In such scenes some artifacts may occur due to the lack of information required for correct handling of partial occlusion, and this is an inherent limitation of the post-filtering approach. We, however, argue that such high-degree blurring is very rare in VR applications. Suppose that we use a projection display located at 3 m in front of the user's pupil, and that its resolution and dimension are 1024×768 and 3.2 m × 2.4 m, respectively. Using the typical lens parameters of the human eye [84], we have the following numbers: d_r=24 mm, E=4 mm, F=17 mm, d_s=3000 mm, d_s/d_r=125, and DPI=0.32 pixels/mm. Considering the shape of the CoC function [53], objects producing maximum CoCs are mostly located at the nearest distance from the eye. If the nearest distance is 300 mm (d=300 mm), the maximum CoC diameter computed by (4.4) does not exceed 10 pixels. This indicates that our DOF framework can be used for VR applications without quality issues.

Another expected advantage of our DOF rendering is related to the depth cue in a VE. A DOF effect in an image is regarded as a relative depth cue rather than an absolute depth cue [69], similarly to other pictorial depth cues (e.g., relative size, texture gradient, and occlusion). In particular, perceiving the relative order of objects is greatly facilitated by the DOF effects [64, 68]. However, sharply maintained foreground boundaries in the previous post-filtering techniques often make this judgment ambiguous. The correct judgment of the relative order requires foreground boundaries to be blurred over focused objects [64]. Since our method uses different blurring behaviors in the background and foreground by smoothing foreground boundaries, we anticipate that our method can help the correct judgment of the relative depth order of objects located in a VE.

Our current framework does not support various point-spread functions (PSFs; also called the intensity distribution functions [18]) of a lens. The PSF of a lens controls the appearance distorted by the aperture shape or the aberration of a lens (called the "bokeh" [70]). The bokeh effect can be realized by stochastic sampling that follows a specific PSF [14], but our framework only supports Gaussian PSF. This is because our method approximates the spatial filtering using coarser images that do not allow stochastic sampling. This limitation, however, is common in the pre-blurring (using downsampled images) or iterative filtering techniques that approximate spatial convolution.

Chapter 5

Real-Time Depth of Field Rendering using Point Splatting on Per-Pixel Layers

This chapter presents a real-time high-quality DOF rendering algorithm that provides the convincing partial occlusion effect without other major artifacts such as the intensity leakage. Two representative examples rendered by our method are shown in Figure 5.1. To obtain the missing information on the occluded area, an additional pinhole image is rendered and combined with the visible image in a way to create proper partial occlusion effects. The composition with the hidden image requires a layered representation such as the scatter method [81] (also called the splatting) or the composition of discrete layers [8, 53]. However, the two methods have some difficulties to be directly used for our purpose. Since the scatter method requires costly *depth sorting* of entire pixels, its real-time use is still challenging. The layer methods well approximate the depth sorting using a few discrete layers, but have another quality problem called the *discretization* artifact [6]. This artifact refers to a band-like artifact appearing in objects separated across multiple layers that do not reflect local depth variation.

Our algorithm is a GPU-based method that combines the advantages of the scatter and layered approaches. Instead of globally defined layers, our method defines three per-pixel

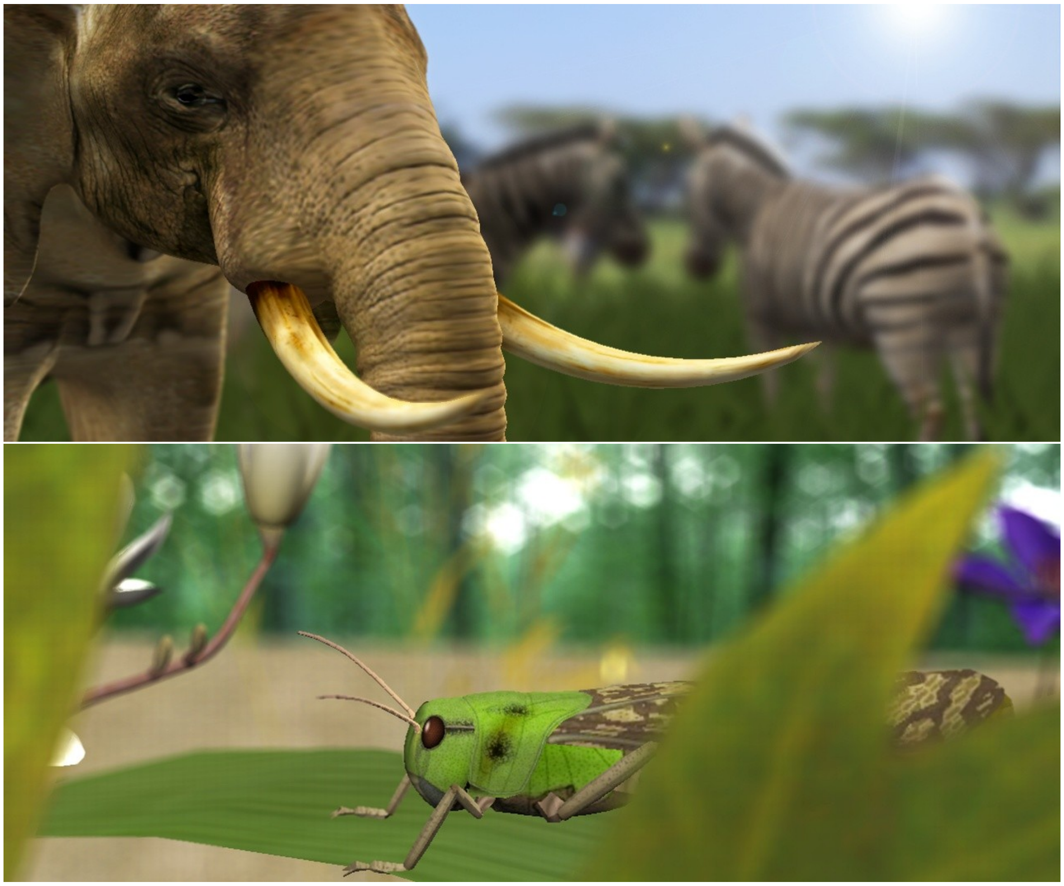


Fig. 5.1 Two example images generated by our DOF rendering method. The upper image shows no intensity leakage, and the lower image shows a partial occlusion effect with defocused highlights of a hexagonal aperture.

layers according to the depth order of a source pixel (scattered from a pinhole image) relative to the destination pixel (to be blurred). This allows even a long object to be decomposed into coherent layers. The visible and hidden images are separately splatted to the per-pixel layers, and then properly blended. As a consequence, our algorithm can produce a high-quality partial occlusion effect. In particular, a sharply focused area occluded by the foreground is *semitransparently* represented with its *accurate colors*.

Another feature of our framework is the amenability to the simulation of photorealistic

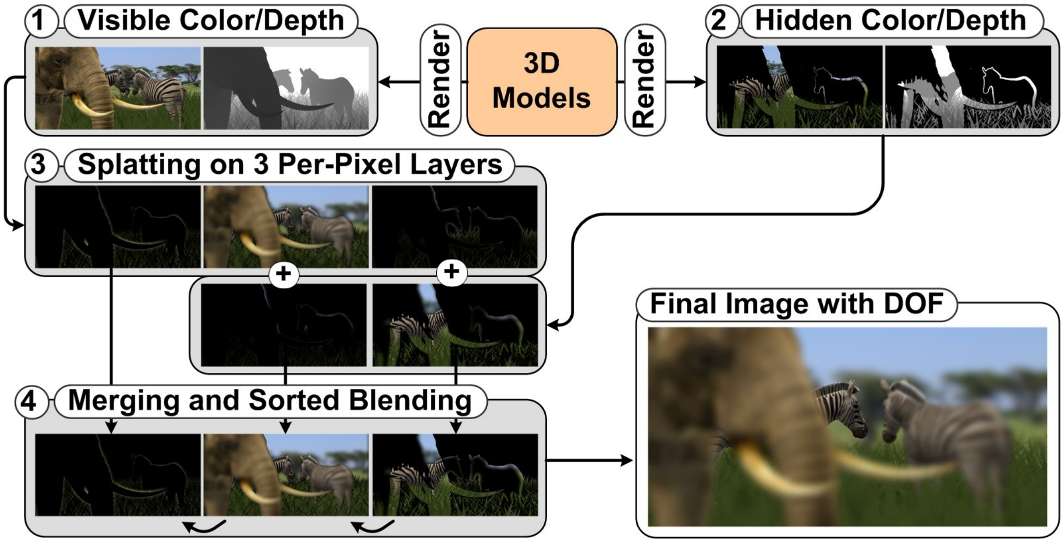


Fig. 5.2 Overview of our DOF rendering framework.

bokeh (distortion of a defocused area) and *defocused highlights* (specular highlights standing out more from blurred areas). These effects can be easily created using the texturing extension of GPU point sprites.

Major contributions of this chapter can be summarized as: (1) a new DOF rendering algorithm using the per-pixel layer decomposition, (2) a method to generate the partial occlusion effect using an occluded pinhole image, (3) a real-time GPU implementation of the intensity scatter algorithm, and (4) its application to the bokeh and defocused highlights.

5.1 Depth-of-Field Rendering Algorithm

The procedure of our DOF rendering framework at run-time is illustrated in Figure 5.2. The circled numbers represent independent steps running on separate GPU programs. First, given a 3D scene, a typical pinhole color/depth image, V, is rendered to a floating-point RGBZ texture using a special GPU program that simultaneously renders color and depth. Second, another color/depth image, H, is rendered for partially occluded areas using another GPU program (Section 5.1.2). Third, the two pinhole images are separately blurred using the point splatting (Section 5.1.3). During the splatting, the source pixels of each sprite are written to one of the three layers according to its relative order for the destination pixel depth (Section 5.1.4). In the last step, the blurred layers are alpha-blended and normalized in the descending order of layer depths (Section 5.1.5), and then the final image is obtained. The subsequent sections present more details.

5.1.1 Relating Depth to CoC

We first describe how to compute a CoC diameter from the depth of each pixel, which is frequently used in the subsequent computations. The computation of the CoC diameter, C, follows the same procedure as that described in Section 4.1.2. The only difference is that we do not store C to a texture. Instead, the CoC diameter is computed on demand.

5.1.2 Rendering of Partially Occluded Area

In addition to a normally rendered visible color/depth image (V), we render an additional color/depth image (H) that contains pixels occluded by the visible pixels. This image is greatly useful for producing semitransparent sharp areas behind the blurry foreground. In typical model rendering, hidden surfaces are overwritten due to the depth test in the fixed rendering pipeline. Thus, in order to render the hidden surfaces, a special fragment shader is required, which is similar to the typical fixed-pipeline shader but performs three additional depth tests using V as follows.

Let Z_v be the depth of a pixel in V, and Z' be the depth of a fragment being processed by the new shader. First, a fragment close enough to the visible pixel ($Z' < Z_v + \epsilon$) is rejected. Since we use a single hidden image, a small offset, ϵ, is controlled such that the fragments belonging to the same object are rejected but those belonging to the second nearest object should pass the test for avoiding artifacts. In our implementation, ϵ=0.05 (on the normalized scale) worked well for most cases. Second, if the visible pixel is in the background ($Z_v > Z_f$), the fragment is rejected, because partial occlusion is scarcely perceived in the background. Finally, a completely hidden fragment is rejected, which incurs redundant splatting overhead. If a depth change in the neighborhood of a visible

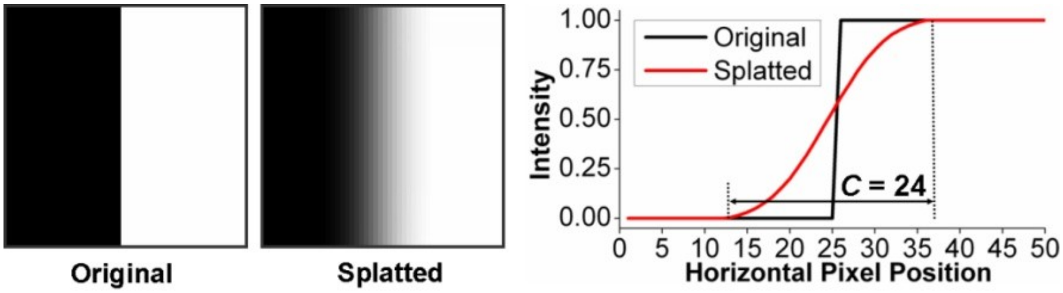


Fig. 5.3 Smooth intensity decrease in the splatted image using CoCs of a 24-pixel diameter for the boundary of the white rectangle. The distance from the edge to the inner area of the full intensity is roughly same to the radius of the CoC.

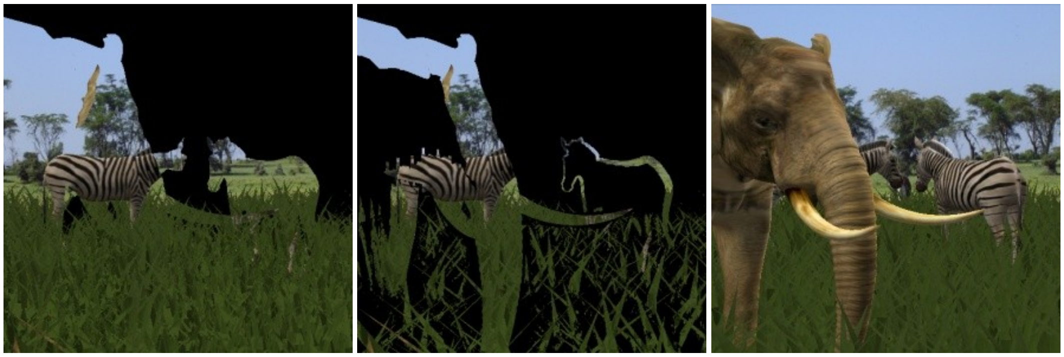


Fig. 5.4 Accepted pixels after the first and second depth tests (left), and the third test (middle) for the Elephant scene (right). The farthest zebra is focused.

pixel (roughly a CoC; see Figure 5.3) is very small, the splatted intensity in the pixel is saturated to one. In that case, the occluded fragment becomes completely invisible due to alpha blending (explained later in Section 5.1.5).

In order to detect the depth change around a pixel position, $\mathbf{p}$, the following operator is applied:

$$\mathcal{L}(\mathbf{p}) = \sum_{\mathbf{n}\in\Omega}(Z_v(\mathbf{n}) - Z_v(\mathbf{p}))/N_\Omega, \tag{5.1}$$

where Ω is a set of neighbor pixels, and N_Ω is the number of elements in Ω. This operator is exactly the same as the Laplacian filter (for edge detection), but, instead of typical 4 neighbors, we use random samples within the CoC of $\mathbf{p}$ (e.g., 12 Poisson disk samples

[21]). If $\mathcal{L}(\mathbf{p})$ is less than a small threshold (e.g., 0.01 on a normalized scale), the fragment is rejected as completely hidden. Although some fragments accepted in this test may be invisible, they are also removed later during the final blending. Figure 5.4 shows the example images of pixels accepted with the consecutive tests.

5.1.3 Image-Space Point Splatting

This section describes how V and H are blurred using the point splatting. The point splatting is a common technique to render a sprite for a 3D point using point sprites present in the GPU, which enlarges a point to a multi-pixel square. The texturing extension of point sprites allows drawing of a circular CoC, even for complex IDFs for bokeh effects.

Given an image, each pixel is mapped to a point in a screen-aligned point array whose dimension is the same as that of the image. When these pixels are enlarged to the size of their CoCs, the overlapped colors stand for the blurred colors.

Due to the scatter of a single pixel on a multi-pixel circle, its intensity should be decreased by its area such that the sum of intensities is equal to one. This intensity decrease is reflected to the alpha channel of the pixel in a sprite (note that the pixel format of a sprite is RGBA unlike RGBZ of V and H). Let $\mathbf{p}$ be the source pixel of a sprite, and $\mathbf{q}$ be the destination position to be written. The alpha value at $\mathbf{q}$, $A(\mathbf{p},\mathbf{q})$, is proportional to the reciprocal of the CoC area with the diameter, $C(\mathbf{p})$, as:

$$A(\mathbf{p},\mathbf{q}) = k_n/C(\mathbf{p})^2, \quad \text{where } \|\mathbf{q}-\mathbf{p}\| \leq C(\mathbf{p})/2. \tag{5.2}$$

k_n is a scaling constant determined by an IDF. For example, a uniform IDF (ideal IDF [18]) uses $k_n{=}4/\pi$. In order to apply various bokeh patterns, IDFs are stored into textures and read back in the fragment shader.

The problem here is that whereas $C(\mathbf{p})$ is defined in a continuous range, a sprite is rasterized into discrete pixels, relying on specific GPU implementation. The left image in Figure 5.5 shows examples of such rasterization patterns when the point of intensity one is splatted for each discrete C (from 1 to 9). Thus, instead of directly using $k_n{=}4/\pi$, we need to compute $k_n(C)$ for each C such that the sum of pixel intensities is one (see the right

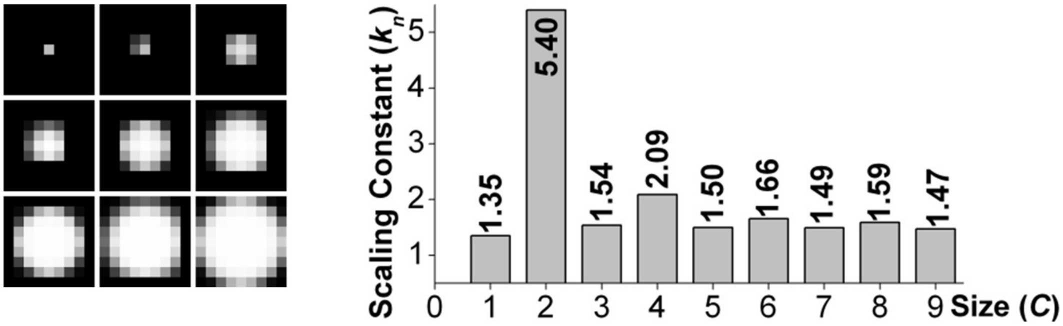


Fig. 5.5 Example rasterization patterns and scaling constants in terms of a point size for the uniform IDF.

graph in the figure). The $k_n(C)$ values are stored in a look-up table (1-D texture) prior to execution, and are read back in the vertex shader at run-time.

Each overlapping sprite is blended using simple accumulative alpha blending. In OpenGL, alpha blending factors are given by GL_SRC_ALPHA, ONE, ONE, ONE for source color, source alpha, destination color, and destination alpha, respectively. Since the single layer splatting blurs a focused boundary and the hidden pixels cannot be represented, we extend the process to the per-pixel layered splatting.

5.1.4 Splatting on Per-Pixel Layers

A layer decomposition technique considering local depth variation is explained in this section. In general, local depths around a pixel have the following properties: (1) partial occlusion occurs around where adjacent depths of a pixel abruptly change, and (2) the depths are concentrated on a few narrow ranges in most cases. Based on these observations, we group source pixels of similar depths in the incoming sprites to three layers for each destination pixel. When the layers are properly blended (Section 5.1.5), the partial occlusion and sharply focused boundary are effectively handled. Due to relying on local depth variation rather than global layers [8, 53], our method is mostly free of the discretization problem, and the reduced layers can generate the effects similar to those by a number of global layers.

More specifically, an incoming source pixel from **p** is assigned to one of the three—in front of (l=0), at the same depth (l=1), or behind (l=2) the visible pixel at **q**—layers, as

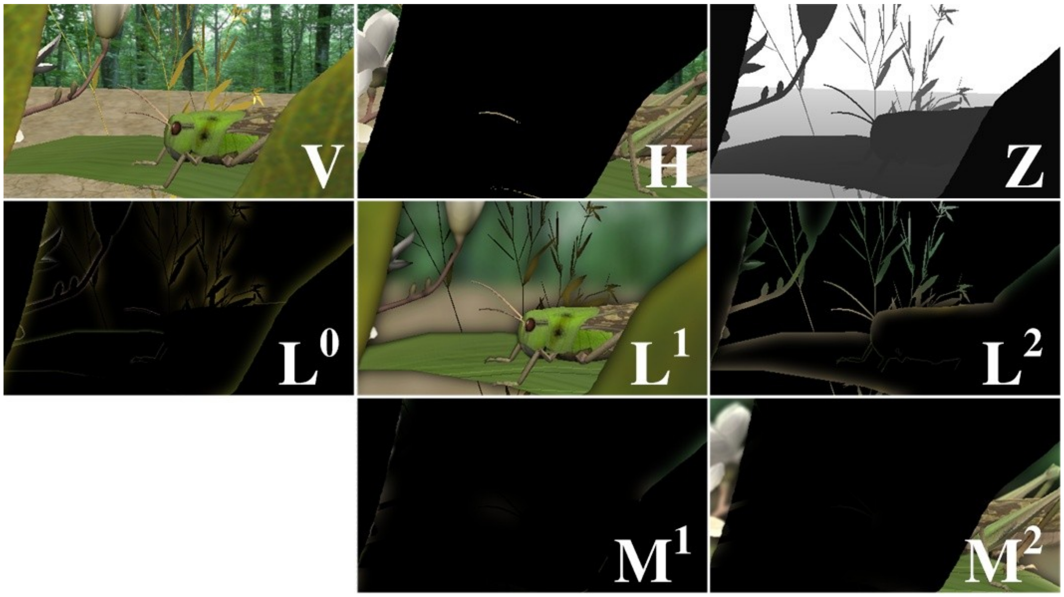


Fig. 5.6 An example of the layer decomposition.

follows.

$$l(\mathbf{p},\mathbf{q}) = \begin{cases} 0 & B(\mathbf{p}) - B_v(\mathbf{q}) < -\delta C_v(\mathbf{q}) \\ 1 & |B(\mathbf{p}) - B_v(\mathbf{q})| \leq \delta C_v(\mathbf{q}) \\ 2 & B(\mathbf{p}) - B_v(\mathbf{q}) > \delta C_v(\mathbf{q}) \end{cases}, \tag{5.3}$$

where B is a signed version of C (i.e., $B{=}K(Z{-}Z_f)/Z$, $K{=}DPI(d_s/d_r)EF/(Z_f{-}F)$). $B(\mathbf{p})$ is computed from the image currently being processed (V or H), and $B_v(\mathbf{q})$ and $C_v(\mathbf{q})$, from V. δ is a constant for determining whether the incoming pixel from $\mathbf{p}$ belongs to the same depth group as $\mathbf{q}$. These conditions smartly control the decomposition according to the blurring degree at $\mathbf{q}$; that is, if $C_v(\mathbf{q})$ is relatively small, layers are strictly decomposed, but otherwise are loosely decomposed. Regarding the use of δ, δ that is too small assigns the pixels belonging to the same object to different layers, which causes the discretization problem. Conversely, δ that is too large assigns most pixels to a single layer, resulting in no partial occlusion. For most examples in this chapter, empirically chosen δ ($0.3 \sim 0.5$) worked well.

According to the index, l, each processed fragment for V is written to one of the three layer images, L^0, L^1, or L^2, and, similarly for those of H, to M^1 or M^2. For H, M^0

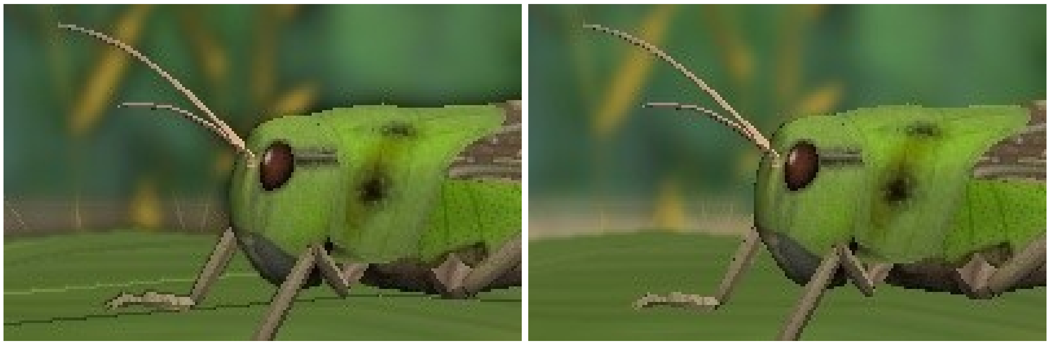


Fig. 5.7 Example images blended without (left) and with (right) normalization. Darker and brighter regions in the left image were correctly normalized in the right image.

is not defined, because hidden pixels do not occlude any pixels. This separate writing is implemented using the multiple-render-target capability in the GPU. Figure 5.6 shows example images of the splatted layers. We can observe that the portions of sprites affecting neighbor pixels are well written to the separate layers, even for depth variations in a small area.

Note that we consider only three layers for real-time implementation. Hence, if more than three depth groups are present around a pixel, a hidden area may not be well represented. However, this problem rarely occurs in practice, unless many thin objects overlap together (see Figure 5.11, for example). If not constrained by a real-time use, this problem can be accurately resolved by additional depth scatter for finding multiple depth layers (possibly, with clustering of the scattered depths).

5.1.5 Composition of Layers and Normalization

The final step for the DOF rendering is the composition of the blurred layers. The composition follows typical sorted blending from farther to nearer layers [8, 53]. Prior to composition, M^1 and M^2 are simply added to L^1 and L^2, respectively, because they are mostly in similar depths. Then, the images are blended via the following three steps.

$$\begin{array}{rcl} D & \leftarrow & L^2 \\ D & \leftarrow & L^1 + (1 - L^1_a)D \\ D & \leftarrow & L^0 + (1 - L^0_a)D \end{array} , \tag{5.4}$$

where D is the accumulating image, and L_a^0 and L_a^1 are the alpha components of L^0 and L^1, respectively. Here, D in each step has "hyper" and "hypo" intensities [72] due to the overlapped CoCs of various sizes (see bright/dark regions in Figure 5.7). Thus, non-zero pixels in each D are divided by their alpha so that each alpha value becomes one. Figure 5.7 shows a result of the normalization process.

5.2 Defocused Highlights

Additional feature of our DOF rendering is the simulation of defocused highlights that occur due to the high intensity contrast in real illumination. This effect plays an important role for artistic photorealism, with various bokeh shapes. However, it cannot be well represented in a typical LDR image. In a LDR image, the incoming light intensity exceeding the maximum pixel value (e.g., 1) is truncated to the maximum, and thus the intensity of the resulted CoC becomes smaller than that of the original CoC.

Thus, in order to attain correct defocused highlights, a non-truncated HDR image is required. If so, rendering of the effect is straightforward, similarly to the case of the motion blur with a HDR image [25]. Instead of splatting on the truncated image, the splatting is applied on the HDR image, and then the resulting image is tone-mapped to an LDR image. Figure 5.8 shows example images. Reinhard tone-mapping [82] ($a = 0.72$) was applied before and after splatting for the left and right images, respectively.

Furthermore, although an LDR image lacks accurate intensities, we can mimic the effect using a common trick that extrapolates intensities of bright areas, as described in [55]. For a pinhole LDR image, the color of each pixel, **F**, is extrapolated by the successive operations as follows.

$$\begin{array}{rcl} \lambda & = & 0.3\ F_r + 0.59\ F_g + 0.11\ F_b \\ \lambda_n & = & (\lambda - \lambda_0)/(1 - \lambda_0) \\ \mathbf{F} & \leftarrow & ((1 - \lambda_n^\beta) + \lambda_n^\beta \gamma)\ \mathbf{F} \end{array} , \tag{5.5}$$

where λ is the luminance of **F**, computed by red, green, and blue components: F_r, F_g, and F_b. λ_0 is the threshold for expanding luminance, and λ_n is the renormalized luminance. β is the exponent for falloff, and γ is the expanding gain. This operation is applied only when

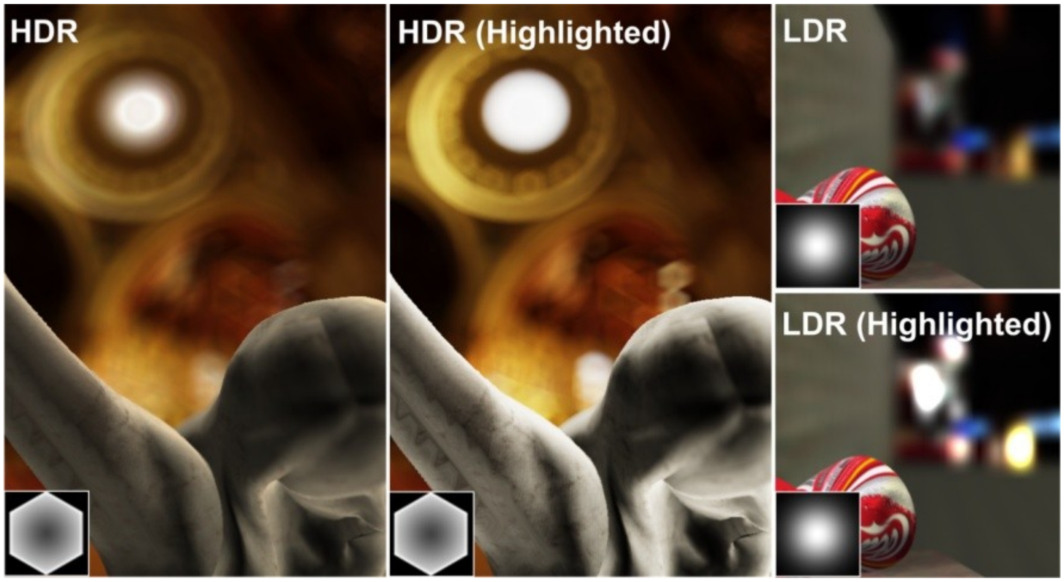


Fig. 5.8 Example results with and without defocused highlights rendered with HDR and LDR images. The bottom-left portion in each image represents the IDF texture used (hexagonal and Gaussian IDFs). Each highlighted image shows better intensity contrast.

$\lambda > \lambda_0$. Figure 5.8 also shows example images generated by this technique. In the figure, the parameters were set as λ_0=0.8, β=3, and γ=2.

5.3 Acceleration of Rendering Performance

The most pressing obstacle to real-time implementation of our method is the heavy data transfer from the vertex processor to numerous fragments. Unlike the gather approach, the scatter structure cannot exploit a stochastic sampling scheme, because a point should be mapped to the whole sprite. Instead, by reducing the resolution of the point array, rendering performance can be improved.

However, a simple reduction in resolution may result in a magnification artifact around focused regions (e.g., a CoC less than 4 pixels) or where partial occlusion occurs. If we reduce the resolution (e.g., 1/4 of the original) except for such areas, it can be a good alternative of the naive rendering with little perceptual degradation. Since we already have

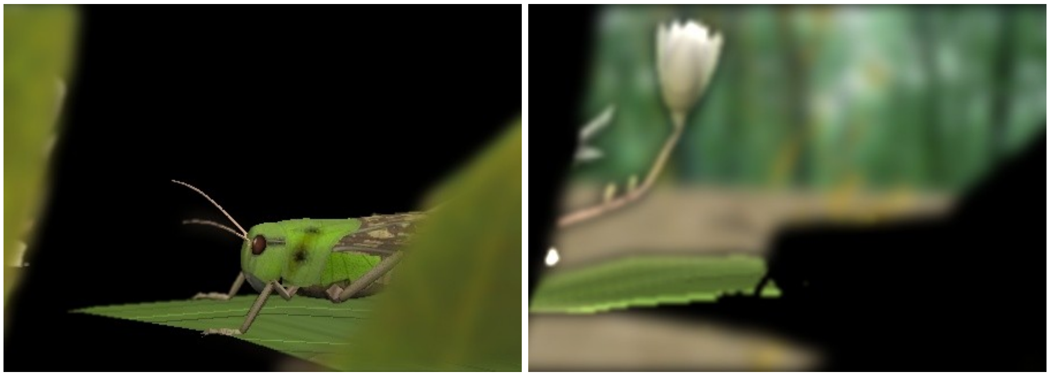


Fig. 5.9 Rendered images at the original (left) and reduced (right) resolutions. Normalization was not applied.

information related to partial occlusion (H) and CoCs, such pixels can be easily found.

In addition, we note that the blurring difference between the two resolutions may cause small holes around overlapping edges. This can be resolved by using one more hidden image for the reduced version, which is similar to H, but the scanning range for $\mathcal{L}$ operator is slightly (e.g., 5%) reduced. This additional hidden image can be simply achieved using H instead of model rendering. Accordingly, the two versions are slightly overlapped, and the artifacts are removed. Figure 5.9 shows an example. Foreground boundaries and focused regions are blurred at the original resolution, but the rest, at 1/4 reduced resolution.

The remainder of this section reports on the measured rendering performance of our methods with and without the acceleration, a recent GPU-based method [103], and the accumulation buffer method [43]. We have implemented the four methods using OpenGL and OpenGL Shading Language on a Pentium 2.67GHz Dual-Core with a GeForce 9800GX2. The test was conducted at a 1024×768 resolution, for the Elephant (216,080 triangles) and Grasshopper (46,009 triangles) scenes shown in Figure 4.1. We used the mean CoC size in an image as a control variable. For the accumulation buffer method, 24, 96, and 384 Poisson disk samples were used for obtaining similar blurring degrees.

Table 5.1 shows the benchmark result. Our method significantly improved the performance over the non-accelerated method. Zhou's method [103] showed superior perfor-

Table 5.1 A performance benchmark for the four methods in terms of frame rate for Elephant (E) and Grasshopper (G) scenes. Subscripts represent mean CoC sizes.

Method	E_8	E_{16}	E_{32}	G_8	G_{16}	G_{32}
Accelerated	67	50	31	68	30	17
Not Accelerated	14	4	1	16	5	3
[103]	105	82	57	144	105	67
[43]	12	3	.7	15	4	.9

mance as well. The frame rate of the accumulation buffer method rapidly decreased along CoC sizes (i.e., number of model rendering). According to the result, the accelerated version of our method can be used at least up to average 16 pixels of CoC diameters for real-time applications. We note that 16-pixel average of CoCs can represent a larger variation (e.g., up to a CoC diameter of a maximum 24 pixels).

5.4 Results and Discussion

In this section, we report and discuss the quality of images rendered by our method. Two representatives were already shown in Figure 4.1. Overall, our method generated natural DOF effects for both foreground and background, without noticeable artifacts. The upper image shows no intensity leakage, similarly to the recent methods [10, 49, 103]. The lower image shows impressive defocused highlights of the hexagonal aperture shape.

With respect to the foreground blurring, we compared our results with Zhou's method (see Figure 5.10) in terms of the peak signal-to-noise ratio (PSNR). The PSNR was computed as $10\ \log_{10}(255^2/MSE)$. The image rendered by the accumulation buffer method was used as a reference. With our result (using a hidden image), a reasonably high PSNR (36.33 dB) was achieved, whereas Zhou's method and ours (not using a hidden image) showed rather low PSNRs (29.4 and 32.84 dB, respectively). This mostly comes from the difference of foreground representation. At a glance, our result generated with a hidden image is very similar to the reference image, whereas the other two show discontinuous boundaries in the foreground (see the magnified portions). Even with our method, a convincing foreground cannot be achieved without the hidden image (actually, similar to Potmesil's

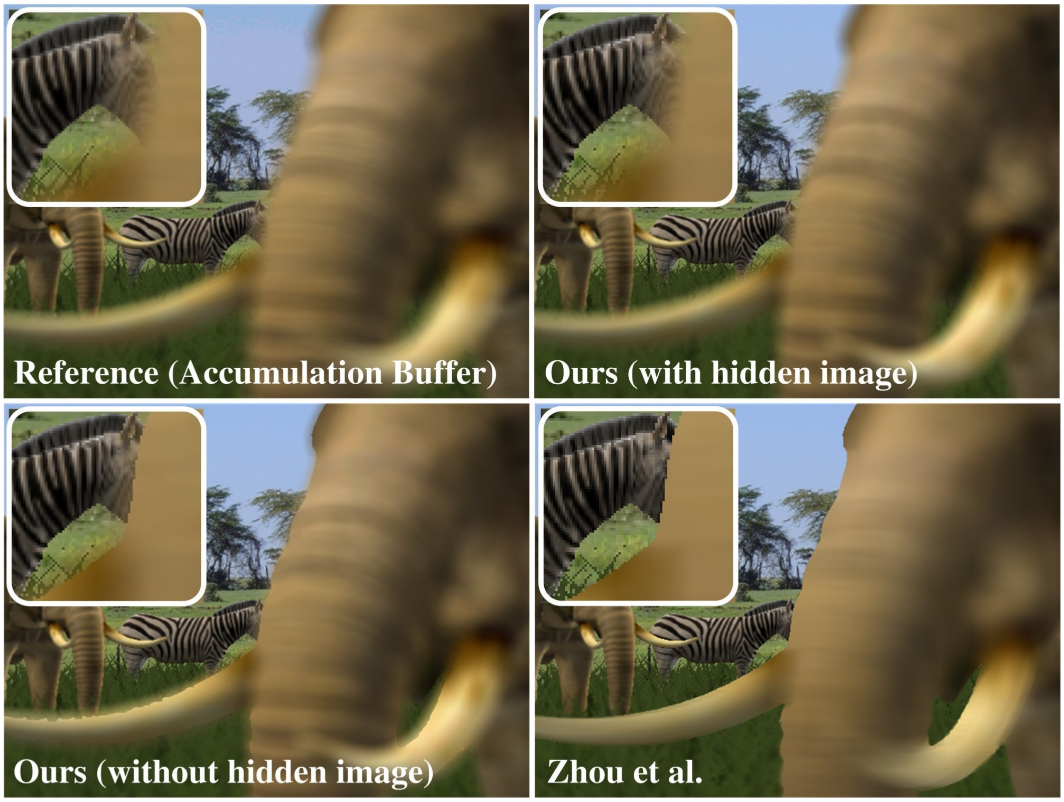


Fig. 5.10 A comparison of DOF rendering results by our method and those of [103] and [43]. The rounded boxes represent the magnified portions in the images for comparison of foreground blurring.

original approach [81]).

Recently, several other methods have also reduced blurring discontinuity in the foreground [6, 49, 53]. However, it seems that they may work only for relatively less blurred scenes that can be covered by extrapolation. Another fundamental problem of these methods is blurring of the focused area to remove sudden blurring changes around the extrapolated boundary, whereas our method successfully maintains the sharply focused area. These problems are similarly applied to the blurring of depth images [10, 29], another simpler method against the blurring discontinuity.

In order to assess our results for multiple-layer cases, we provide a case-by-case compar-

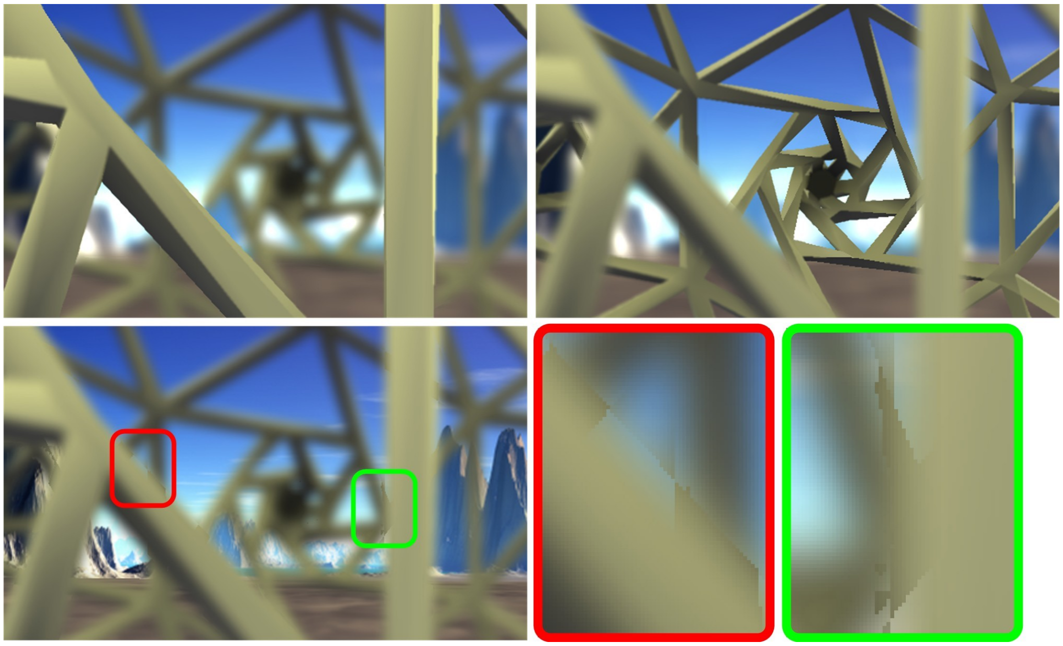


Fig. 5.11 Example images rendered for a scene focused on the near (top left), middle, (top right), and far (bottom left) objects. Whereas the first two are well represented, the third exhibits small artifacts (see the red and green boxes magnified in the bottom right side).

ison for a scene where thin geometries overlap together (see Figure 5.11). Three cases—focused on near, middle, and far objects—were compared. Whereas the first and second cases were well represented, small artifacts (see red boxes) are observed for the third case (the multiple foreground objects of different depths exist around a focused area). This problem results from the limitation of our layer decomposition and hidden image; we only use the second nearest hidden image and a single foreground/background layer. As already mentioned, this problem can be resolved using exact depth groups and multiple hidden images, but with longer computational time. Nevertheless, this is not a severe problem in practice, compared to the discontinuity in the foreground boundaries.

As a final example, Figure 5.12 demonstrates the results when our method is applied to a scene with long objects. Since our method well decomposes even a long object into coherent layers, the discretization artifact is not observed in the figure (see hand rails in the left

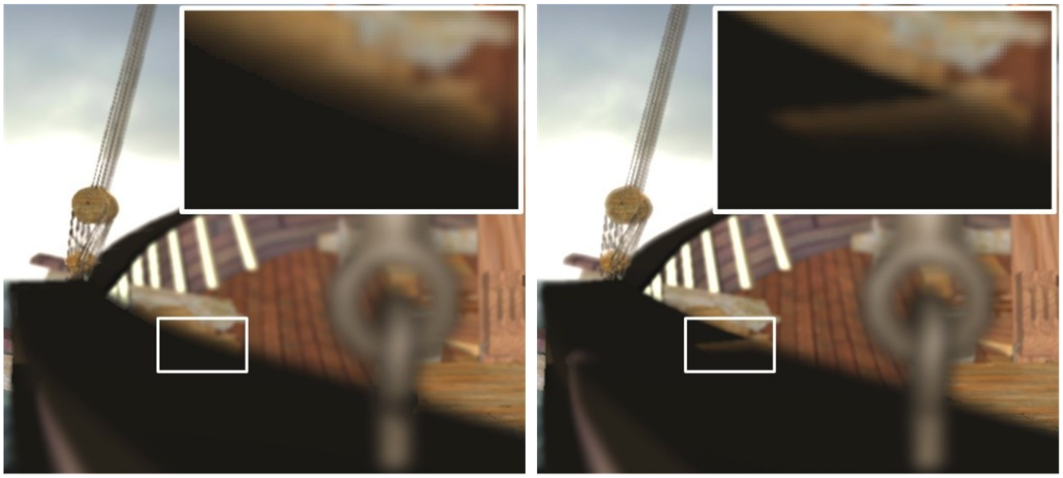


Fig. 5.12 Two example images generated using per-pixel layers (left) and globally defined layers (right) for a scene with long objects.

image). However, such a result might be hard to achieve, if a long object would be allowed to be separated across multiple layers. The right image rendered using globally defined layers [8] shows an example. This is because the insufficient (less than one) intensities at the layer boundaries can make the colors of a hidden area permeate the boundaries during alpha blending (similarly to the "black band" problem [6]).

Lastly, we summarize the current limitations of our framework. First, our method requires floating-point texturing capability. However, the difference of acceleration performance for 8-bit and 16-bit textures is practically marginal with an aid of recent GPUs (supporting shader model 4.0). Second, the anti-aliasing is not supported, which is a common problem in most post-filtering methods due to the use of a depth buffer that cannot be anti-aliased. The last limitation is the need of manual tuning of parameters. The performance of our method mainly depends on ϵ (in Section 5.1.2) and δ (Equation 5.3). We used ϵ=0.05 and δ=0.4 for most examples in this chapter, and tested them for various scenes. Whereas these simple parameters work quite well for moderately blurred scene, a much blurred (e.g., a CoC diameter greater than 32 pixels) scene requires slight tuning of the parameters, according to the spatial configuration of a scene.

Chapter 6

Attention-Guided Depth-of-Field Rendering

DOF rendering is one of the applications where perceptual image quality can be improved with our visual attention tracking framework. This chapter explains how DOF rendering methods, described in Chapters 4 and 5, can be integrated with the attention tracking framework. The combined rendering system shows improved perceptual quality of a visual scene with real-time rendering performance enough to be used in interactive VEs.

6.1 Integration of Attention Tracking to Depth-of-Field Rendering

A procedure of rendering DOF effects with the attention tracking is straightforward. For every rendering frame at runtime, the depth of an object predicted to be the most attentive is used as a focal depth, Z_f. Since we already know the depth of the attended object (used in the computation of spatial context), no additional computation is required for the depth.

However, since our attention tracking framework selects only foreground objects (excluding backdrops such as sky, wall, and floor), two minor problems are encountered: (1) the discontinuous change of a focal depth along the line of sight and (2) the case where no foreground objects are visible in a scene. The first problem arises from the fact that our attention tracking framework does not consider saccadic eye movements[1], while the smooth pursuits can be considered as an analogy of the smooth tracking of a single attended object. Thus, in-between gaze movements where an attended target is changed to another one are not defined. Such saccadic eye movements can be simulated by interpolating the depths of the objects either in image space or 3D space. Since DOF simulation depends solely on the focal depth, the focal depth is interpolated in the 1-D depth space for our purpose. As for the second problem, when there are no visible foreground objects in the scene, we use the depth picked from the center of the screen as a focal depth. More details are presented in what follows.

Interpolation of Focal Depths in Depth Space

In general, a saccadic eye movement accompanies both of reflexive and voluntary ocular motor controls [66]. The reflexive ocular motor control is closely related to common jitter patterns to stabilize the vision independently of a head movement. Since our framework does not incur such instability, we only need to simulate voluntary movements to locate region of interest (ROI) into fovea. In classic eye-tracking methodology, voluntary movements in the raw eye-tracking data are detected using an approximate linear model [27], which is an operational simplification of the underlying nonlinear natural process [15]. Similarly to the linear model, we simulate the saccadic eye movements by temporally interpolating the depths of previous and next fixated objects. The temporal transition begins when the attended object (with the highest attention value) is switched to another. The duration of the transition controls the smoothing degree of the focus change. In our implementation, we set the duration as 300 ms, based on that the latency of refocusing for misfocused vergence is approximately 200-300 ms [32, 65].

Given the depths of the previous and next attended objects, Z_p and Z_n, the focal depth, Z_f, is computed using temporal blending:

$$Z_f(t) = lerp(Z_p(t), Z_n(t), t/\tau), \tag{6.1}$$

---

[1] Saccadic eye movement refers to rapid shifts of the line of sight of the two eyes from one target to another for locating objects of interest to a fovea. In contrast, smooth pursuit refers to the smooth tracking behavior on objects of interest [27].

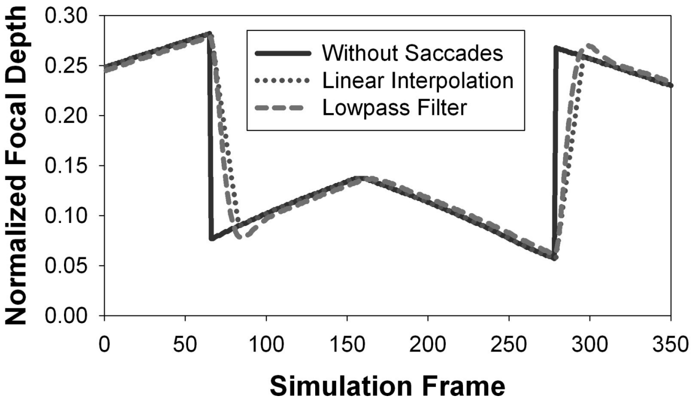


Fig. 6.1 Examples of saccades simulation using the linear interpolation and lowpass filter.

where t ($\in [0, \tau]$) is a time period elapsed from the beginning of the transition, τ is the duration of transition (e.g., 300 ms), and $lerp(x, y, z)$ is the typical linear interpolation function. We note that Z_p and Z_n are also varied along the transition time; that is, the focused objects can move during the transition.

This linear interpolation method works well for most cases, but it cannot handle a very quick change of fixated objects; for instance, a focus change that occurs before completing the current transition often yields a temporal delay. In order to deal with such a rapid change, low-pass filtering can be employed as an alternative to the linear interpolation. In the low-pass filtering, the smoothing degree is controlled by a cut-off frequency. Our empirical test suggests that cut-off frequency below than 2 Hz is adequate.

Fig. 6.1 shows the examples of saccadic eye movements simulated by the two methods, linear interpolation and lowpass filter. The linear interpolation used the duration of 300 ms, and lowpass filter used a cut-off frequency of 1 Hz. The test uses the Elephant scene shown in Figure 6.2 and the rendering method presented in Chapter 4. To generate abrupt changes of a focal depth, we moved between the forward and backward repeatedly, and made the attended object alterenated between the elephant and the zebra. As can be seen in Fig. 6.1, the linear interpolation method well suppressed the abrupt change of the focal depth, and the focal depth smoothly changed along the transition time. The lowpass filter method produced a locus similar to the linear interpolation, but exhibited overshoot effects

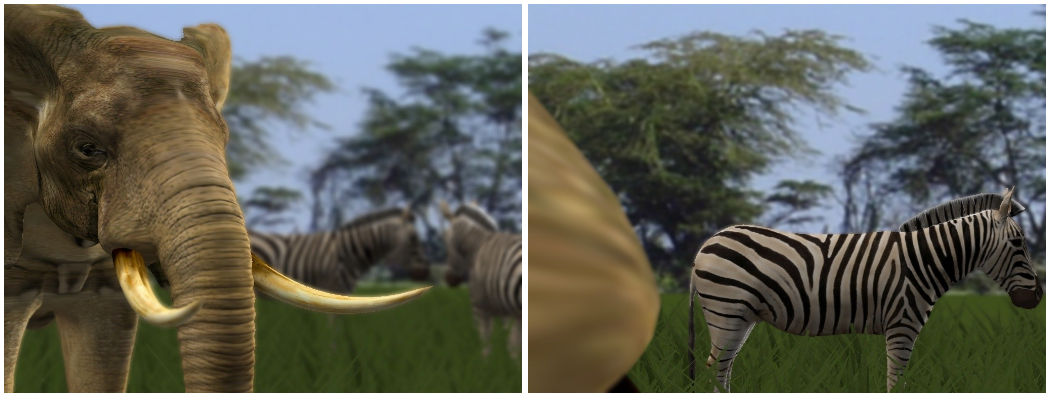


Fig. 6.2 A test scene used for simulating saccadic eye movements. The focus was altered between the elephant and zebra.

at the ending of the transition. The lowpass filter, instead, can handle the quick change of the focus, while the linear interpolation method cannot respond to such a change until the prior transition is completed.

Handling Absence of Foreground Objects

The remaining issue is handling the case where no targets to be focused are defined in the scene. In this case, we need to determine the focal depth only from the background. In pilot experiments, we observed that the gaze of a user usually tends to return to the center of the screen, if no targets to be fixated are found in the scene. Based on this observation, when the attention value of a single foreground object falls under a certain threshold and thereby the object becomes invisible, the depth picked in the center of the screen is used as a focal depth. The depth of the screen center is used as either a previous or next focal depths for the interpolation (described above).

6.2 Performance Evaluation

We have measured rendering performance of the proposed attention-directed DOF rendering framework in terms of frame rate. The framework was implemented on a 2.67 GHz Pentium Core2Duo PC with a GeForce 9800GX2 graphics card. A 1024 × 768 display res-

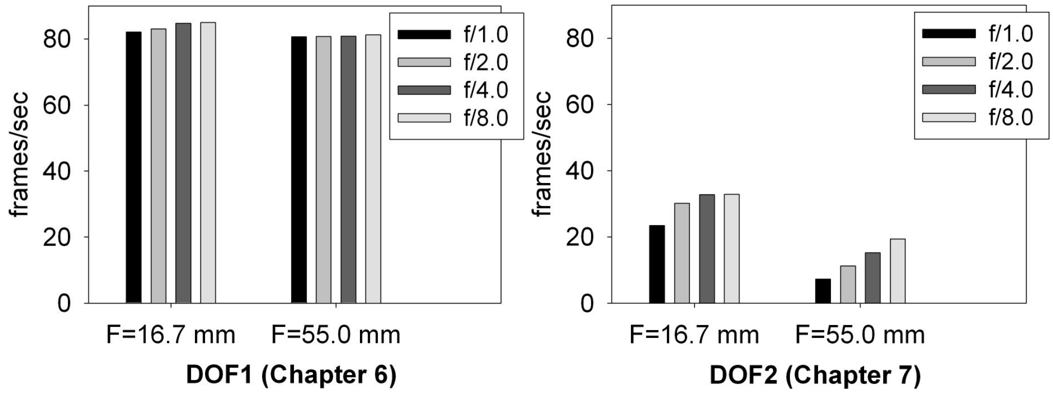


Fig. 6.3 Comparison of measured frame rates for rendering DOF effects with the visual attention tracking by the methods presented in Chapters 4 and 5.

olution and a 64 × 64 saliency map were used in the test. The elephant scene with 456,114 triangles, shown in Fig. 6.2, was used for the test. Unlike the controlled DOF rendering, interactive DOF rendering cannot directly control the blurring degree, since the amount of blurring depends on the focal depth as well as the optical properties of a lens (e.g., aperture size and focal length). The degree of blurring was varied by the focal length and f-number (=F/E). We used two focal lengths, $F = 16.7$ mm and $F = 55$ mm, which are the typical values of the human visual system and the photography, respectively. Four f-numbers, 1.0, 2.0, 4.0, 8.0, were used. The two DOF rendering methods (denoted by DOF1 and DOF2, which were presented in Chapters 4 and 5, respectively) were used in the test.

Fig. 6.3 shows the benchmark results. For DOF1, we can observe that rendering performances under all conditions are sufficient to be used in interactive VR applications (at least more than 80 frames/sec), and the performances are mostly independent of the amount of blurring. As for DOF2, the rendering performance significantly relies on the amount of blurring. For $F = 16.7$ mm, the performance was approximately more than 25 frames/sec, implying that the framework can be used for interactively simulating the DOF effects in the human retina. On the other hand, the performances under $F = 55.0$ mm (greatly blurred) were inappropriate for the interactive applications. In such a case, DOF2 can be used only for static scene.

6.3 Discussions

This section discusses the expected advantages and limitations of the attention-guided DOF rendering compared to the naive rendering based on a pinhole lens model and the typical DOF rendering based on the eye-tracking, respectively. We also note current limitations of the implemented framework.

When a virtual scene is rendered with DOF effects guided by visual attention, there is no doubt that such a rendering significantly improves the visual realism, helps quickly locate the egocentric or exocentric frames of references by differentiating focused objects from background/foreground, and thereby, reduces the mental load required for visual search. In addition to those benefits, DOF rendering is known to mediate monocular (pictorial) depth perception in the human visual system [64, 67, 68, 76, 69, 96]. Despite the general agreement on the positive role of DOF effects on depth perception in pictures or photographs, whether it can help depth perception in interactive VEs has not been thoroughly examined. This is primarily due to the fact that defocused blur depends on focal configuration, in addition to the depths of objects, which can be dynamically varied along a user's eye movements. However, since most previous studies have paid attention to static images with a fixed focal configuration, their results cannot be easily applied to dynamic focusing in VR. Moreover, since a typical VE commonly presents occlusion and motion parallax, the true effects of DOF blur as a depth cue should be investigated by considering their collective effects. In our future work, the effects of DOF blur towards depth perception will be investigated in depth.

Over the previous methods based on eye-tracking devices (e.g., [45]), the attention-guided DOF rendering has two advantages: (1) stable rendering without trembling of the gaze and (2) more accurate prediction of the attended objects in the presence of an interposition of objects. Despite the recent development of high-performance eye-tracking devices, most eye-tracking devices still exhibit some trembling or flickering, because the human eyes are tracked in coarser resolution due to the limited processing power of image sensors and CPU. On the other hand, since our attention-guided DOF rendering does not use a physical image sensor, it does not result in such a temporal instability. The second advantage results from the limitation of monocular eye tracking. In a typical VR system, since the accommodation is fixed on the flat display, eye-tracking uses a monocular tracking, making the measurement of a 3D depth impossible. Thus, such an eye-tracking has an innate ambiguity in selecting a gazed object in the presence of interposition of objects. The ambiguity can yield significantly incorrect tracking, possibly with motion sickness. In comparison, our attention-guided DOF rendering less suffers from the problem, because the temporal context of visual attention tracking can maintain an attended object to be followed even in the presence of partial occlusion. Also, our object-based scene graph reduces the probability of wrong prediction in singling out the attended object.

Finally, the current implementation of the attention-guided DOF rendering exhibits two limitations. First, our framework inherently involves a tracking delay, because it uses the user's past interaction. Although this delay can adversely affect the task performance, depth perception, and visual realism of rapid navigation, it is not a severe problem in practice for slow navigation. The second problem is the relatively lower prediction accuracy compared to the eye-tracking. This arises from the fact that our method cannot predict the gaze unrelated to the user's interaction; that is, if a user moves his/her eyes without navigation or deliberately looks at the side opposite to the moving direction, our framework cannot correctly predict the accurate gaze. Nonetheless, under the common interaction behavior, our framework can produce plausible results to be used in practice.

Chapter 7

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 chapter, 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 Estimation 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 run-time execution. However, they have a common perceptual problem, ‘popping’ effect, which is induced from the discrete model transitions between adjacent LOD levels. In this chapter, we adapt ‘Unpopping’ LOD (ULOD) recently proposed by Giegl et al. [38], where one model of an object is opaquely rendered and the other model is semi-transparently 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 off-line pre-processing step, we simplify an original object model into a few multi-resolution models. Then, for every rendering frame at run-time, 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 they 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 proposed to use computational heuristics that approximate the expected perceptual benefit by considering various user-defined factors such as image coverage (or size), semantic importance, focus, motion, and simplification accuracy [36, 40]. 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 in [36, 40], 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 [40, 61] as:

$$cost_{NOLOD} = \mathcal{T}_n + \mathbf{t} \cdot \mathbf{1}, \tag{7.1}$$

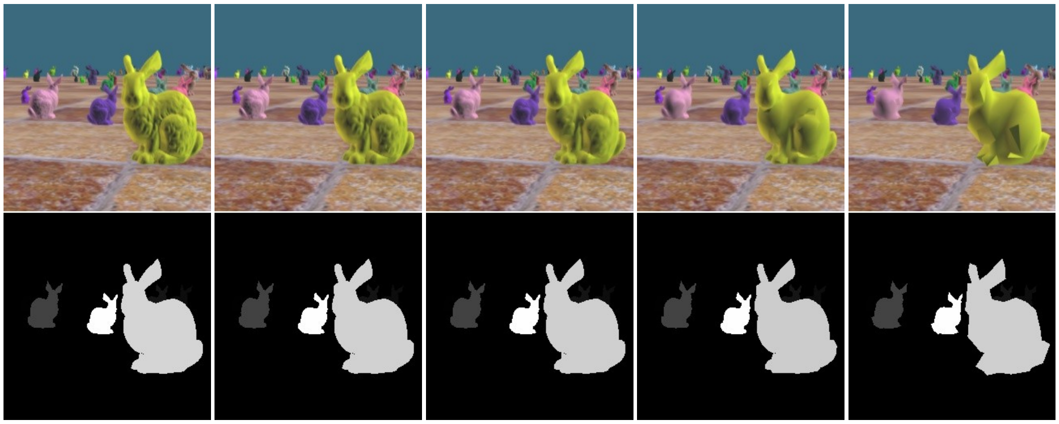
where $\mathcal{T}_n$ is the time required for initialization, finalization, and set-up of objects, $\mathbf{t}$ is the vector of the rendering times of original object models, and $\mathbf{1}$ is a 1-vector. The rendering cost including LOD management with attention tracking can be expressed as:

$$cost_{LOD} = \mathcal{T}_n + \mathcal{T}_a + \mathbf{t} \cdot (\mathbf{a} + \mathbf{r}). \tag{7.2}$$

Here, $\mathcal{T}_a$ 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). $\mathbf{a}$ 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 $\mathbf{r}$ is a simplification degree that ranges from 1 (the original) to 0 (a no-polygon model).

In order to obtain performance gain, the cost gain by the use of LOD must be greater than the additional cost required for the attention estimation such that $\mathbf{t} \cdot (\mathbf{a} + \mathbf{r}) \ll \mathbf{t} \cdot \mathbf{1}$. 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 $\mathbf{a}$), although $\mathcal{T}_a$ is practically negligible for a complex scene (as shown in Fig. 3.5, 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 $\mathbf{a}$ are close to 2.

Thus, given a scene, in order to reduce the rendering cost of the feature maps, we suggest two techniques. Firstly, 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 $\mathbf{a}$ close to 1. Nevertheless, the item/depth buffer still require rendering of models. Secondly, 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 $\mathbf{a}$ to be much smaller than $\mathbf{1}$ and $\mathbf{r}$. For instance, if we render the coarsest model with 1/256 triangles of the original model, each entry in $\mathbf{a}$ 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.



(a)L_0: 1 (b)L_1: 1/4 (c)L_2: 1/16 (d)L_3: 1/64 (e)L_4: 1/256

Fig. 7.1 Comparison between the object attention maps rendered using the raw and simplified models. The upper images are rendered color images, and the lower images, the corresponding object attention maps. The triangle numbers of simplified models were 1/4, 1/16, 1/64, and 1/256 of the original one, respectively. The Bunny model was provided through the courtesy of Stanford 3D Scanning Repository.

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. 7.1 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 difference among all the pairs were statistically insignificant (all p-values $<$ .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

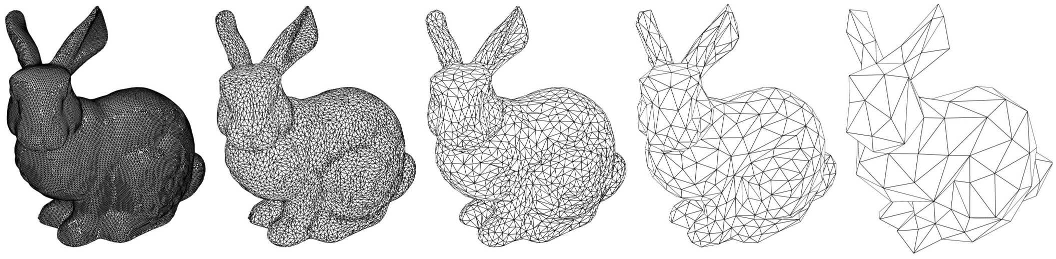
For LOD management, the level of an object model to be rendered needs to be determined among its multi-resolution models. The model level can be associated with the attention value of the object in a straightforward manner. Let $M(k)$, $L(k)$, and $r(k, L)$ be the highest level (the coarsest resolution) 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:

$$L = M(1 - S_o). \tag{7.3}$$

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 multi-resolution 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:

$$r(L) = \beta^{-L}, \tag{7.4}$$



(a)L_0 (b)L_1 (c)L_2 (d)L_3 (e)L_4

Fig. 7.2 Simplified Bunny models for LOD management. The original model consists of 69,451 triangles. The triangle numbers of the simplified models decrease by a quarter scale from left to right (69,451, 17,364, 4,341, 1,086 and 272 triangles, respectively).

where β 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 [37], which has been widely used for mesh simplification. Fig. 7.2 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 Chapter 3.5. 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 is 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. 7.3 shows examples of the Dragon scene used in the experiment.

We measured the rendering cost (time) during free navigation of the scene. Since all the

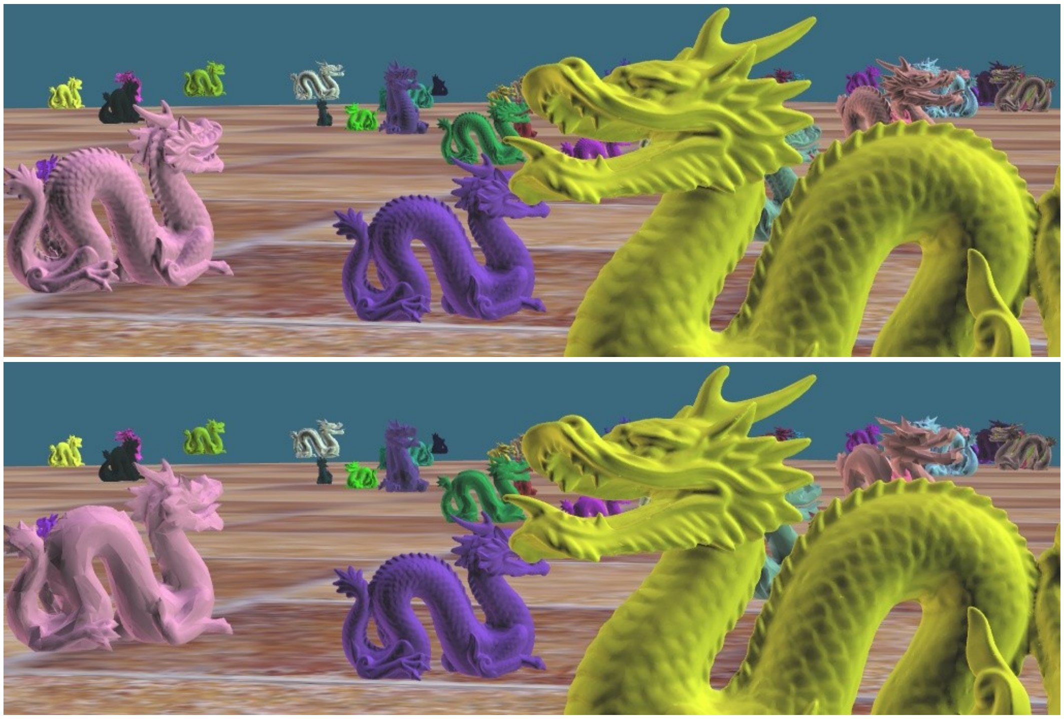


Fig. 7.3 Two example images rendered without (upper) and with (lower) LOD management. In the lower image, non-attentive objects were simplified based on their object attention values. The Dragon model was provided through the courtesy of Stanford 3D Scanning Repository.

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 (ULOD2, ULOD3, ULOD4, and ULOD5). For instance, ULOD3 represented ULOD 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

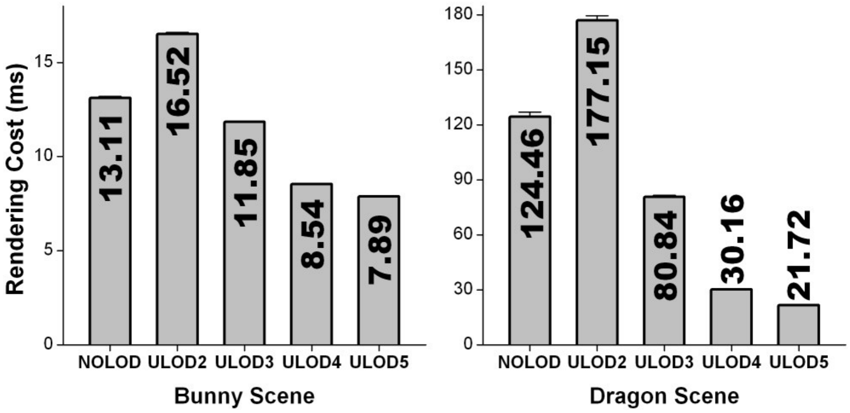


Fig. 7.4 Mean rendering costs under NOLOD, ULOD2, ULOD3, ULOD4, ULOD5 in the Bunny and Dragon scenes. The vertical error bars represent the standard errors.

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. 7.4. Overall, as expected, the rendering costs of NOLOD were higher than those of the ULODs, except for ULOD2. Compared to the Bunny scene, the rendering costs for the Dragon scene were more reduced. For both scenes, ULOD4 and ULOD5 showed high performance improvements.

Fig. 7.5 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 ULOD conditions. As the number of objects gradually increased, all ULOD methods outperformed NOLOD. However, the performance gain of ULOD2 required significantly many objects (roughly more than 40 objects). For the Dragon scene, ULOD3, ULOD4, and ULOD5 exceeded NOLOD, even for scenes with a small number of objects, whereas ULOD2 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 con-

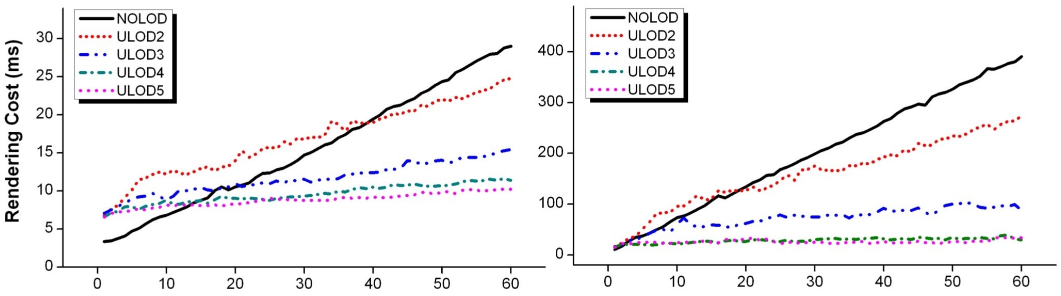


Fig. 7.5 Evolution of the rendering costs with respect to the number of visible objects in the Bunny (left) and Dragon (right) scenes. The triangle number of a non-simplified Bunny was 69,541. The triangle number of a Dragon was 435,708.

ventional 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. 7.5, 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 expect that this limitation can be partially surmounted by utilizing only top-down contextual information without bottom-up saliency. As already shown in Fig. 3.10, 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.

Chapter 8

Conclusion and Future Work

In this dissertation, the author presented a comprehensive, practical, and real-time platform for perceptual rendering in interactive and dynamic VEs. Our visual attention tracking framework allows more accurate prediction of visually attended objects by utilizing user's volitional factors as well as conventional feature-driven saliency map. The user study on the estimation accuracy showed that the newly included top-down contextual information played a pivotal role in singling out the most plausible candidate to be attended. The implemented GPU-based framework is suitable for real-time use, even if considering additional costs required for LOD management or DOF rendering. The author also proposed the two DOF rendering methods better than the previous real-time methods. Our methods successfully achieved high-quality DOF effects as well as remarkable real-time performance.

As perceptual rendering applications, we demonstrated attention-guided DOF rendering and LOD management as representatives of applications for improving perceptual quality and rendering performance, respectively. The LOD management integrated with the attention tracking framework is a better performance regulation metric to encompass a user's intention beyond the conventional feature-driven passive metrics. The attention-guided DOF rendering exhibited perceptually improved image quality with real-time performance appropriate for VR applications as well. The attention-guided DOF rendering is the first attempt for interactive DOF rendering without an eye tracker.

In the future, the author will work on: (1) improving the spatial context model to include the novelty of objects of consideration, (2) developing adequate top-down context models for other common user tasks in VEs, (3) investigating effects of defocus blur on perceptual benefits such as perceptual realism and salient distinction between focused and blurred objects, and (4) developing a more efficient and accurate method to render the DOF effect.

요 약 문

가상환경에서의 관심 추적 및 실시간 지각 기반 렌더링

본 논문은 가상 환경에서 시각적 관심의 계산적인 추적을 기반으로 하는 실시간 지각 기반 렌더링 체계를 제안한다. 시각적 관심 추적 시스템은 기존의 특징 기반 샐리언시 맵 외에 가상환경 내 내비게이션으로부터 추론한 사용자의 의도를 반영하여 가장 관심을 받을만한 객체를 실시간에 찾아낸다. 구현된 체계의 예측 정확도를 눈동자 추적 장치를 이용하여 채집한 사람의 눈동자 움직임과 비교/평가하기 위해 사용자 실험을 수행하였고, 그 결과는 인식 이론에 의해 잘 설명되는 정확도를 보여준다. 관심 추적 체계는 지각적 품질과 성능 향상을 위한 대표적인 두 방법인 상세도 관리와 필드심도 렌더링에 적용되었다. 필드심드 렌더링에의 적용에 앞서, 인터랙티브 가상환경에 적합한 기존 방법이 없었기에, 저자는 GPU 기반의 두 가지 실시간 필드심도 렌더링 방법을 제안한다. 첫 번째 방법은 기존의 밉맵 기반 접근 방식을 확장하고, 다른 방법은 기존의 레이어/분산 방식을 확장한다. 두 방법 모두 기존 방법들에서 보이는 이미지 단점을 효과적으로 제거하고, 실시간 사용에 적합한 성능을 보인다. 제안한 두 필드심도 렌더링 방법과 함께, 저자는 관심 기반의 필드심도 렌더링과 상세도 관리를 제안한다. 두 방법은 관심을 받는 물체의 깊이와 관심 정도를 초점 깊이와 상세도로 각각 이용한다. 관심 기반의 필드심도 렌더링은 눈동자 추적 장치 없이 인터랙티브한 렌즈 블러 효과를 시뮬레이션하고, 관심 기반의 상세도 관리는 지각적으로 품질 저하 없이 렌더링 성능을 대폭 향상한다.

Bibliography

[1] S. T. Acton. Edge enhancement of infrared imagery by way of the anisotropic diffusion pyramid. In *Proc. Image Processing*, pages 865–868. IEEE, 1996.

[2] T. Akenine-Möller, J. Munkberg, and J. Hasselgren. Stochastic rasterization using time-continuous triangles. In *Proc. Graphics Hardware*, pages 7–16, 2007.

[3] V. Aurich and J. Weule. Non-linear gaussian filters performing edge preserving diffusion. In *Proc. DAGM-Symposium*, pages 538–545. Springer, 1995.

[4] E. Awh and H. Pashler. Evidence for split attentional foci. *Journal of Experimental Psychology*, 26(2):834–846, 2000.

[5] G. Backer, B. Mertsching, and M. Bollmann. Data- and model-driven gaze control for an active-vision system. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 23(12):1415–1429, 2001.

[6] B. Barsky, M. Tobias, D. Chu, and D. Horn. Elimination of artifacts due to occlusion and discretization problems in image space blurring techniques. *Graphical Models*, 67:584–599, 2005.

[7] B. A. Barsky. Vision-realistic rendering: Simulation of the scanned foveal image from wavefront data of human subjects. In *Proc. 1st Symposium on Applied perception in graphics and visualization*, pages 73–81, 2004.

[8] B. A. Barsky, A. W. Bargteil, D. D. Garcia, and S. A. Klein. Introducing vision-realistic rendering. In *Proc. Eurographics Rendering Workshop*, pages 26–28, 2002.

[9] A. K. Beeharee, A. J. West, and R. J. Hubbold. Visual attention based information culling for distributed virtual environments. In *Proc. ACM Symposium on Virtual Reality Software and Technology*, pages 213–222, 2003.

[10] M. Bertalmío, P. Fort, and D. Sánchez-Crespo. Real-time, accurate depth of field using anisotropic diffusion and programmable graphics cards. In *Proc. 3D Data Processing, Visualization, and Transmission*, pages 767–773. IEEE Computer Society, 2004.

[11] K. Bjorke. High-quality filtering. In R. Fernando, editor, *GPU Gems: Programming Techniques, Tips & Tricks for Real-Time Graphics*, chapter 24, pages 391–415. Addison-Wesley Professional, 2004.

[12] D. E. Broadbent. *Perception and Communication*. Pergamon, London, 1958.

[13] R. Brown, L. Cooper, and B. Pham. Visual attention-based polygon level of detail management. In *Proc. the 1st international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, pages 55–ff, Jan. 2003.

[14] J. Buhler and D. Wexler. A phenomenological model for bokeh rendering. In *ACM SIGGRAPH abstracts and applications*, 2002.

[15] R. H. S. Carpenter. *Movements of the Eyes*. London: Pion, 1977.

[16] K. Cater, A. Chalmers, and P. Ledda. Selective quality rendering by exploiting human inattentional blindness: Looking but not seeing. In *Proc. ACM Symposium on Virtual Reality Software and Technology*, pages 17–24, 2002.

[17] K. R. Cave and N. P. Bichot. Visuospatial attention: Beyond a spotlight model. *Psychonomic Bulletin & Review*, 6(2):204–223, 1999.

[18] Y. C. Chen. Lens effect on synthetic image geneartion based on light particle theory. *The Visual Computer*, 3(3):125–136, 1987.

[19] J. H. Clark. Hierarchical geometric models for visible surface algorithms. *Communications of the ACM*, 9(10):547–554, 1976.

[20] C. E. Connor, H. E. Egeth, and S. Yantis. Visual attention: Bottom-up versus top-down. *Current Biology*, 14(19):R850–R852, Oct. 2004.

[21] R. L. Cook. Stochastic sampling in computer graphics. *ACM Trans. Graphics*, 5(1):51–72, 1986.

[22] R. L. Cook, T. Porter, and L. Carpenter. Distributed ray tracing. *Computer Graphics*, 18(3):137–145, 1984.

[23] S. M. Culhane and J. K. Tsotsos. An attentional prototype for early vision. In *Proc. European Conference on Computer Vision*, pages 551–560, 1992.

[24] J. Cutting and P. Vishton. Perceiving layout and knowing distances. In W. Epstein and S. Rogers, editors, *Perception of Space and Motion*, pages 69–117. Academic Press, San Diego, CA, 1995.

[25] P. E. Debevec and J. Malik. Recovering high dynamic range radiance maps from photographs. *Proc. ACM SIGGRAPH*, pages 369–378, 1997.

[26] J. Demers. Depth of field in the 'Toys' demo. In *Game Developers Conference*, 2003.

[27] A. T. Duchowski. *Eye Tracking Methodology: Theory and Practice*. Springer, 2007.

[28] J. Duncan. Selective attention and the organization of visual information. *Journal of Experimental Psychology. General*, 113(4):501–517, 1984.

[29] J. Earl Hammon. Practical post-process depth of field. In H. Nguyen, editor, *GPU Gems 3: 3D and General Programming Techniques for GPUs*, chapter 28, pages 583–606. Addison-Wesley, 2007.

[30] S. Engel, X. Zhang, and B. Wandell. Colour tuning in human visual cortex measured with functional magnetic resonance imaging. *Nature*, 388(6637):68–71, 1997.

[31] J. T. Enns. Three-dimensional features that pop out in visual search. In Taylor and Francis, editors, *Visual Search*, pages 37–45. London: Taylor & Francis, 1990.

[32] J. T. Enright. Facilitation of vergence changes by saccades: influences of misfocused images and of disparity stimuli in man. *Journal Physiology*, 371:69–87, 1986.

[33] B. Erickson and F. Romano. *Professional Digital Photography*. Prentice Hall, 1999.

[34] P. Fearing. Importance ordering for real-time depth of field. In *Proc. Third International Computer Science Conference, Hong Kong*, volume 1024 of *LNCS*, pages 372–380. Springer, 1996.

[35] C. L. Folk, R. W. Remington, and J. C. Johnston. Involuntary covert orienting is contingent on attentional control settings. *Journal of Experimental Psychology. Human Perception and Performance.*, 18(4):1030–1044, 1992.

[36] T. A. Funkhouser and C. H. Séquin. Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments. In *Proc. ACM SIGGRAPH*, pages 247–254, 1993.

[37] M. Garland and P. Heckbert. Surface simplification using quadric error metrics. *Proc. ACM SIGGRAPH*, pages 209–215, 1997.

[38] M. Giegl and M. Wimmer. Unpopping: Solving the image-space blend problem for smooth discrete LOD transitions. *Computer Graphics Forum*, 26(1):46–49, 2007.

[39] D. Gillham. Real-time depth-of-field implemented with a postprocessing-only technique. In W. Engel, editor, *ShaderX*^{5}*: Advanced Rendering Techniques*, chapter 3.1, pages 163–175. Charles River Media, 2006.

[40] E. Gobbetti and E. Bouvier. Time-critical multiresolution scene rendering. In *Proc. IEEE Visualization*, pages 123–130, 1999.

[41] E. B. Goldstein. *Sensation and Perception. 5th edition.* Brooks/Cole Publishing Company, 1998.

[42] J. Haber, K. Myszkowski, H. Yamauchi, and H.-P. Seidel. Perceptually guided corrective splatting. *Computer Graphics Forum*, 20(3), 2001.

[43] P. Haeberli and K. Akeley. The accumulation buffer: Hardware support for high-quality rendering. *Proc. ACM SIGGRAPH*, pages 309–318, 1990.

[44] J. M. Henderson. Human gaze control during real-world scene perception. *Trends in Cognitive Sciences*, 7(11):498–504, 2003.

[45] S. Hillaire, A. Lécuyer, R. Cozot, and G. Casiez. Using an eye-tracking system to improve camera motions and depth-of-field blur effects in virtual environments. In *Proc. IEEE Virtual Reality*, pages 47–50, 2008.

[46] H. Hoppe. Progressive meshes. *Proc. ACM SIGGRAPH*, pages 99–108, 1996.

[47] L. Itti, C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Trans. Pattern Analasis and Machine Intelligence*, 20(11):1254–1259, 1998.

[48] D. J. Jobson, Z. ur Rahman, and G. A. Woodell. Properties and performance of a center/surround retinex. *IEEE Trans. Image Processing*, 6(3):451–462, 1997.

[49] M. Kass, A. Lefohn, and J. Owens. Interactive depth of field using simulated diffusion on a GPU. Technical report, Pixar Animation Studios, 2006.

[50] J. Kessenich, D. Baldwin, and R. Rost. The OpenGL Shading Language. 3Dlabs, Inc. Ltd., 2004.

[51] C. Koch and S. Ullman. Shifts in selective visual attention. *Human Neurobiology*, 4:219–227, 1985.

[52] T. J. Kosloff and B. A. Barsky. An algorithm for rendering generalized depth of field effects based on simulated heat diffusion. In *Proc. Computational Science and Its Applications*, pages 1124–1140, 2007.

[53] M. Kraus and M. Strengert. Depth-of-field rendering by pyramidal image processing. *Computer Graphics Forum*, 26(3), 2007.

[54] B. Kuipers. Modeling spatial knowledge. *Cognitive Science*, 2:129–153, 1978.

[55] H. Landis. Production-ready global illumination. ACM SIGGRAPH Course Note, 2002.

[56] C. H. Lee, A. Varshney, and D. W. Jacobs. Mesh saliency. *ACM Trans. Graphics*, 24(3):659–666, 2005.

[57] Y. LeGrand. *Light, Color and Vision, 2nd Edition*. Chapman and Hall, 1968.

[58] G. R. Loftus and N. H. Mackworth. Cognitive determinants of fixation duration during picture viewing. *Journal of Experimental Psychology*, 4:565–572, 1978.

[59] B. London, J. Upton, and B. Brill. *Photography*. Prentice Hall, 2002.

[60] P. Longhurst, K. Debattista, and A. Chalmers. A GPU based saliency map for high-fidelity selective rendering. In *Proc. 4th International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa*, pages 21–29, January 2006.

[61] D. Luebke, B. Watson, J. D. Cohen, M. Reddy, and A. Varshney. *Level of Detail for 3D Graphics*. Elsevier Science Inc., New York, NY, USA, 2002.

[62] D. P. Luebke and C. Erikson. View-dependent simplification of arbtrary polygonal environments. *Proc. ACM SIGGRAPH*, pages 199–208, 1997.

[63] Y. F. Ma, X. S. Hua, L. Lu, and H. Zhang. A generic framework of user attention model and its application in video summarization. *IEEE Trans. Multimedia*, 7(5):907–919, 2005.

[64] J. Marshall, C. Burbeck, D. Ariely, J. Rolland, and K. Martin. Occlusion edge blur: A cue to relative visual depth. *Journal of the Optical Society of America*, 13:681–688, 1996.

[65] P. Maruff, D. Hay, V. Malone, and J. Currie. Asymmetries in the covert orienting of visual spatial attention in schizophrenia. *Neuropsychologia*, 33(10):1205–1223, 1995.

[66] G. Masson, L. Proteau, and D. R. Mestre. Effects of stationary and moving textured backgrounds on the visuo-oculo-manual tracking in humans. *Vision Research*, 35(6), 1995.

[67] G. Mather. Image blur as a pictorial depth cue. *Biological Sciences*, 263(1367):169–172, 1996.

[68] G. Mather. The use of image blur as a depth cue. *Perception*, 26:1147–1158, 1997.

[69] G. Mather and D. R. Smith. Blur discrimination and its relation to blur-mediated depth perception. *Perception*, 31(10):1211–1219, 2002.

[70] H. M. Merklinger. A technical view of bokeh. *Photo Techniques*, 1997.

[71] M. C. Mozer and M. Sitton. Computational modeling of spatial attention. In H. Pashler, editor, *Attention*, pages 341–393. UCL Press, London, 1998.

[72] J. D. Mulder and R. van Liere. Fast perception-based depth of field rendering. In *Proc. ACM Virtual Reality Software and Technology*, pages 129–133. ACM, 2000.

[73] A. L. Nagy and R. R. Sanchez. Critical color differences determined with a visual search task. *Journal of the Optical Society of America*, 7(7):1209–1217, 1990.

[74] K. Nakayama and G. Silverman. Serial and parallel processing of visual feature conjunctions. *Nature*, 320:264–265, 1986.

[75] K. M. O'Craven, P. E. Downing, and N. Kanwisher. fmri evidence for objects as the units of attentional selection. *Nature*, 401(6753):584–587, 1999.

[76] R. P. O'Shea, D. G. Govan, and R. Sekuler. Blur and contrast as pictorial depth cues? *Perception*, 26(5):599–612, 1997.

[77] N. Ouerhani, J. Bracamonte, H. Hugli, M. Ansorge, and F. Pellandini. Adaptive color image compression based on visual attention. In *Proc. ICIAP*, pages 416–421, 2001.

[78] N. Ouerhani, R. von Wartburg, and H. Hugli. Empirical validation of the saliency-based model of visual attention. *Electronic Letters on Computer Vision and Image Analysis*, 3(1):13–24, 2004.

[79] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 12(7):629–639, July 1990.

[80] R. Peters and L. Itti. Beyond bottom-up: Incorporating task-dependent influences into a computational model of spatial attention. *Proc. IEEE Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[81] M. Potmesil and I. Chakravarty. A lens and aperture camera model for synthetic image generation. *Proc. ACM SIGGRAPH*, 15(3):297–305, 1981.

[82] E. Reinhard, M. Stark, P. Shirley, and J. Ferwerda. Photographic tone reproduction for digital images. In *Proc. ACM SIGGRAPH*, pages 267–276, 2002.

[83] G. Riguer, N. Tatarchuk, and J. Isidoro. Real-time depth of field simulation. In W. F. Engel, editor, *ShaderX^2: Shader Programming Tips and Tricks with DirectX 9*, chapter 4, pages 529–556. Wordware, 2003.

[84] P. Rokita. Generating depth-of-field effects in virtual reality applications. *IEEE Computer Graphics and its Application*, 16(2):18–21, 1996.

[85] U. Rutishauser, D. Walther, C. Koch, and P. Perona. Is bottom-up attention useful for object recognition? In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 37–44, 2004.

[86] A. Santella and D. DeCarlo. Visual interest and npr: an evaluation and manifesto. In *Proc. the 3rd International Symposium on Non-Photorealistic Animation and Rendering*, pages 71–150. ACM, 2004.

[87] T. Scheuermann. Advanced depth of field. In *Game Developers Conference*, 2004.

[88] C. Scofíeld. 2.5d depth-of-field simulation for computer animation. In D. Kirk, editor, *Graphics Gems III*, chapter 1.8, pages 36–38. Morgan Kauffman, 1994.

[89] C. Sears and Z. Pylyshyn. Multiple object tracking and attentional processing. *Journal of Experimental Psychology*, 54(1):1–14, 2000.

[90] M. Shinya. Post–filtering for depth of field simulation with ray distribution buffer. In *Proc. Graphics Interface*, pages 59–66, 1994.

[91] A. W. Siegel and S. H. White. The development of spatial representations of large-scale environments. In H. Reese, editor, *Advances in Child Development and Behavior*, volume 10, pages 10–55. Academic Press, New York, 1975.

[92] S. Smith and J. Brady. Susan - a new approach to low level image processing. *International Journal of Computer Vision*, 23(1):45–78, 1997.

[93] F. M. Speed, R. R. Hocking, and O. P. Hackney. Methods of analysis of linear models with unbalanced data. *Journal of the American Statistical Association*, 73(361):105–112, 1978.

[94] A. M. Treisman and G. Gelade. A feature-integration theory of attention. *Cognitive Psychology*, 12:97–136, 1980.

[95] B. Watson, N. Walker, L. F. Hodges, and A. Worden. Managing level of detail through peripheral degradation: Effects on search performance with a head-mounted display. *ACM Trans. Computer-Human Interaction*, 4(4):323–346, 1997.

[96] S. J. Watt, K. Akeley, M. O. Ernst, and M. S. Banks. Focus cues affect perceived depth. *Journal of Vision*, 5:834–862, 2005.

[97] H. Weghorst, G. Hooper, and D. P. Greenberg. Improved computational methods for ray tracing. *ACM Trans. Graphics*, 3(1):52–69, 1984.

[98] G. Welsh and G. Bishop. An introduction to the kalman filter. Technical Report 95-041, Univ. of North Carolina at Chapel Hill, 1995.

[99] J. M. Wolfe. Guided search 2.0. In *Proc. the Human Factors and Ergonomics Society 37th Annual Meeting*, pages 1295–1299, 1993.

[100] J. C. Xia, J. El-Sana, and A. Varshney. Adaptive real-time level-of-detail-based rendering for polygonal models. *IEEE Trans. Visualization and Computer Graphics*, 3(2):171–181, 1997.

[101] A. L. Yarbus. *Eye Movements and Vision*. Plenum Press, New York, NY, 1967.

[102] H. Yee, S. Pattanaik, and D. P. Greenberg. Spatiotemporal sensitivity and visual attention for efficient rendering of dynamic environments. *ACM Trans. Graphics*, 20:39–65, 2001.

[103] T. Zhou, J. X. Chen, and M. Pullen. Accurate depth of field simulation in real time. *Computer Graphics Forum*, 26(1), 2007.

감사의 글

제가 청운의 꿈을 안고 포항에 온 지 벌써 15년이 흘러 정든 포항을 떠나게 되었습니다. 그 동안 많은 분들과 함께 지내어 뜻 깊은 학부, 대학원 생활이 될 수 있었습니다. 먼저, 늦게나마 힘든 연구에 희망을 제게 보여주신 최승문, 김정현 교수님께 무한한 감사의 말씀을 올립니다. 그리고, 대학원/학부 생활 지도와 더불어, 무사히 마칠 수 있도록 도와주신 박찬모 전 총장님, 진왕철 교수님께 깊은 감사를 드립니다. 또한, 익숙치 않은 그래픽스 연구에 많은 도움을 주신 이승용 교수님과, 바쁜 시간을 내셔서 흔쾌히 논문 심사를 해 주신 이인권, 최승진 교수님께 감사를 드립니다.

7년의 대학원 과정을 지내다 보니, 열거하기 힘들 정도로 많은 분의 도움을 받았습니다. 인생의 선배로 물심 양면 도와주신 채종한, 장두현, 남상헌, 김종현, 최유영, 김권동, 이미정, 한미영, 정태호, 박유, 김은희, 김미자, 장혜자, 송보학, 박상준, 황승욱 선생님께 진심의 감사의 말씀을 올립니다. 지금은 졸업하신 선배인 상윤형, 남규형, 진석형, 재인, 건, 재용형, 창석형, 수연형, 용석형, 동식, 정우, 수영, 자연, 선형, 지혜에게 함께 할 수 있어서 감사하단 말씀을 전합니다. 또한, 먼저 나간 동기/후배들, 보현, 태용, 형진, 진욱, 선명, 유진, 광훈, 재영, 재훈, 용진, 채현도 모든 일이 잘 되길 바랍니다. 아직 남아서 연구하는 종현, 석희, 성훈은 남은 시간 힘들더라도 건강하고 바라는 바를 성취하기를 바라고, 인, 인욱, 갑종, 건혁, 재봉에게 모범이 되지 못한 선배로서 미안했고, 모두 앞으로 건강하고 연구에 행운이 함께하기를 바랍니다.

마지막으로, 부족한 절 믿고 기다려주신 부모님, 장인어른, 장모님께 감사의 말씀을 올리면서, 힘든 가운데 끝까지 믿고 따라준 수연에게 사랑한다는 말과 함께 이 논문을 바칩니다.

Curriculum Vitae

Name : Sungkil Lee

Date of Birth : 1975. 11. 6

Present Address : 경북 포항시 남구 효자동 포항공대 대학원아파트 3-705

Education

1994–2002 : B.S. in Materials Science and Engineering, POSTECH

2002–2009 : Ph.D. in Computer Science and Engineering, POSTECH
Thesis Title :
가상환경에서의 관심 추적 및 실시간 지각 기반 렌더링**(Real-Time Perceptual Rendering with Computational Visual Attention Tracking in Virtual Environments)**
Advisor: Prof. Seungmoon Choi

Publications

1. Sungkil Lee, Gerard Jounghyun Kim, and Seungmoon Choi "Real-Time Depth-of-Field Rendering Using Anisotropically Filtered Mipmap Interpolation" *IEEE Trans. Visualization and Computer Graphics*, 2008 (in press).
2. Sungkil Lee, Gerard Jounghyun Kim, and Seungmoon Choi "Real-Time Tracking of Visually Attended Objects in Virtual Environments and Its Application to LOD" *IEEE Trans. Visualization and Computer Graphics*, 15(1): 6–19, 2009.
3. Sungkil Lee and Gerard Jounghyun Kim "Effects of Visual Cues and Sustained Attention on Spatial Presence in Virtual Environments Based on Spatial and Object Distinction" *Elsevier Interacting with Computers*, 20(4–5): 491–502, 2008.
4. Sungkil Lee, Gerard Jounghyun Kim, and Seungmoon Choi "Real-Time Depth-of-Field Rendering Using Splatting on Per-Pixel Layers" *Computer Graphics Forum*, 27(7): 1955–1962, 2008.
5. Jane Hwang, Jaehoon Jung, Jaeyoung Cheon, Sungkil Lee, Seungmoon Choi, and Gerard Jounghyun Kim "Requirements, Implementation and Applications of Hand-held Virtual Reality" *International Journal of Virtual Reality*, 5(2): 59–66, 2006.
6. Sungkil Lee, Gerard Jounghyun Kim, and Seungmoon Choi "Real-Time Tracking of Visually Attended Objects in Interactive Virtual Environments" in *Proc.*

ACM Symp. Virtual Reality Software and Technology (VRST'07), pp. 29–38, 2007 (Best paper nominated).

7. Sungkil Lee and Gerard Jounghyun Kim “Multiple Pseudo-Attention Tracking in Virtual Environments” in *Proc. The 4th Int. Symp. Ubiquitous VR*, pp. 103–104, 2006.
8. Sungkil Lee, Gerard Jounghyun Kim, Albert Rizzo, and Hyungjin Park “Formation of Spatial Presence: By Form or Content?” in *Proc. 7th Annual International Workshop Presence*, pp. 20–27, 2004.
9. Sungkil Lee, Gerard Jounghyun Kim, and Janghan Lee “Observing Effects of Attention on Presence with fMRI” in *Proc. ACM Symp. Virtual Reality Software and Technology (VRST'04)*, pp. 73–80, 2004.
10. 이성길, 김정현, 최승문 “이방성으로 필터링된 밉맵의 보간을 이용한 실시간 필드 심도 렌더링” in *Proc. HCI Korea*, 2008 (Best paper award).
11. 전석희, 김상기, 박건혁, 한갑종, 이성길, 최승문, 최승진 “동작인식 및 촉감제공 게임 컨트롤러” in *Proc. HCI Korea*, 2008 (Best paper nominated).
12. 이성길, 김정현, 최승문 “GPU에서 픽셀 분산을 이용한 실시간 필드 심도 렌더링” *Journal of Digital Entertainment*, 1(1):45–49, 2007.
13. 이성길, 김정현 “Attention Tracking in 3D Virtual Environments” in *Proc. Korea Computer Graphics Society (KCGS)*, pp. 129–134, 2006.