Graphical Models

EDITORIAL ADVISORY BOARD

EDITORS EMERITI

Graphical Models 76 (2014) 115

Contents lists available at ScienceDirect

Graphical Models

journal homepage: www.elsevier.com/locate/gmod

Preface

This issue contains selected papers from the Second Computational Visual Media Conference (CVM), held in Hangzhou, China, in September of 2013. The CVM conferences are intended to draw together cross-disciplinary researchers who are concerned with the computational processing of visual information. With the advance of sensing and imaging techniques, mobile devices, and the Internet, visual media has become ubiquitous. The amount, diversity, and ambiguity of visual data brings great challenges in management, understanding, processing and utilization. Tackling these challenges often calls for integration of novel ideas from different disciplines including computer graphics, geometric computing, computer vision, machine learning, image and video processing, and visualization.

Continuing the success of the first conference, CVM 2013 has attracted 104 submissions from researchers worldwide. Among them, 21 full papers were selected for oral presentation by the International Programme Committee, and 6 outstanding papers were selected for publication in this special issue of GMOD in extended and revised versions.

The first paper presents a novel algorithm to simplify a mesh with motion-awareness, which effectively preserves features that are perceivable in motion and thereby accelerates the rendering of motion blur. The second paper presents a model retrieval system that combines the complementary strengths of global and local shape features of 2D views of 3D models for more robust comparison with the query sketch. The third paper introduces a method for generating 3D relief models from a single image that mimics how artists create hand-crafted bas-reliefs. The fourth paper formulates and solves texture-mapping of a curved surface as a weighted parameterization problem guided by the importance map of the texture, which effectively preserves the shape of the prominent content in the texture. The fifth paper presents one of the first work on semantic classification of human interaction sequences, and it shows promising results using supervised learning on low-level 3D pose features. The last paper presents a facial animation system for mobile devices and features a novel facial shape regression algorithm that is both more efficient than state-of-art algorithms and can robustly handle lighting changes of the performer.

We hope that the readers will enjoy this special issue and also get interested in the other papers accepted by CVM 2013 which will be published in The Visual Computer and the Journal of Computer Science and Technology. We are grateful to all the paper authors and paper reviewers for their contributions.

Guest Editors
Tao Ju
Department of Computer Science and Engineering Ambassador, McDonnell International Scholar Academy Washington, University in St. Louis, One Brookings Drive, St. Louis, MO 63130-4899, USA
E-mail address: taoju@cse.wustl.edu

Hujun Bao
Information Technology, Computer Science Department, Zhejiang University, 38 Zheda Rd, Xihu, Hangzhou, Zhejiang 310027, China

Available online 12 February 2014

http://dx.doi.org/10.1016/j.gmod.2014.02.001

Graphical Models 76 (2014) 116–127

Contents lists available at ScienceDirect

Graphical Models

journal homepage: www.elsevier.com/locate/gmod

Perception-based model simplification for motion blur rendering

Minying Zhang [a,b], Wencheng Wang [a,*], Hanqiu Sun [c], Honglei Han [a,b,d]

[a] State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, China
[b] University of Chinese Academy of Sciences, China
[c] Department of Computer Science & Engineering, The Chinese University of Hong Kong, China
[d] School of Animation and Digital Arts, Communication University of China, China

ARTICLE INFO

Article history:
Received 11 August 2013
Accepted 3 October 2013
Available online 22 October 2013

Keywords:
Human perception
Model simplification
Motion blur rendering

ABSTRACT

Motion blur effects are important to motion perception in visual arts, interactive games and animation applications. Usually, such motion blur rendering is quite time consuming, thus blocking the online/interactive use of the effects. Motivated by the human perception in relation to moving objects, this paper presents simplified geometric models that enable to speedup motion blur rendering, which has not been tracked in motion blur rendering specifically. We develop a novel algorithm to simplify models with motion-aware, to preserve the features whose characteristics are perceivable in motion. We deduce the formula to outline the level of detail simplification by the object moving velocity. Using our simplified models, methods for motion blur rendering can achieve the rendering quality as using the original models, and obtain the processing acceleration mostly. The experimental results have shown the effectiveness of our approach, more acceleration with the larger models or faster motion (e.g. for the dragon model with over a million facets, the motion-blur rendering via hierarchical stochastic rasterization is sped up by over 27 times).

1. Introduction

Motion blur refers to visible streaks generated by the movement of an object or a camera, resulted from the light integration at the imaging devices during a finite exposure time. It is an important cue in the perception of dynamic objects in motion, and more frequent demand for rendering high-quality animated images. In principle, motion blur rendering needs to draw the object at many different but continuous times and then average the results. By such, the time complexity is quite high. Over years, it has been an active research topic in interactive graphics and virtual reality attempted to speed up motion blur rendering. To address the issue, approaches have been studied, such as investigating visibility coherence and motion hints to simplify the light integration [30], representing the moving objects with static geometry to save time-sampling [9], developing sampling techniques via spatio-temporal coherence or hierarchical structures to accelerate ray tracing of objects in motion [6,27], and optimizing the rendering pipeline to efficiently use caches [27] or GPUs [24]. However, to our knowledge, no method has been proposed to efficiently simplify the moving objects for speeding up motion blur rendering, and determine the level of detail simplification in relation to the motion-blur features in rendering.

In this paper, we propose a motion-aware (direction, speed) simplification method to get the simplified models with which motion blur rendering can be accelerated. As our method is orthogonal to existing techniques for motion blur rendering, it can be easily integrated with existing methods for fast rendering. Here, we mainly focus on two issues with respect to moving objects. First, we simplify the object model with the moving direction aware, aiming to reduce the geometric primitives as many as possible while producing the visible features in motion blur similar

* Corresponding author.
E-mail addresses: zhangmy@ios.ac.cn (M. Zhang), whn@ios.ac.cn (W. Wang), hanqiu@cse.cuhk.edu.hk (H. Sun), hanhonglei@sina.com (H. Han).

1524-0703/$ - see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.gmod.2013.10.003

to using the original model. This is based on the observation that the features parallel to the motion direction are blurred and less likely to be observed in the rendered results. Then we deduce a formula to get the level of detail simplification related to the moving speed to simplify the object, by the sensitivity variation map of the human visual system corresponding to the spatial frequency and motion of the object [13]. Afterwards, we examine how to incorporate our algorithms with the rendering pipeline using rasterization or ray tracing, the two popular approaches for motion blur rendering via 3D models, as illustrated in Fig. 1, with our simplified models, the existing methods get acceleration for motion blur rendering, even by an order of magnitude. In general, our main technical contributions are in two aspects:

- An effective approach for producing simplified 3D models especially to speed up motion blur rendering.
- Novel motion-aware algorithms for geometric models to fast perform high quality motion blur rendering.

In the rest of the paper, we briefly outline the most related work in motion blur rendering, level of detail simplification, and perception-based rendering in Section 2. Then, we analyze the motion blur phenomenon and the perception of motion blur in Section 3. Afterwards, we describe direction-aware simplification with object motion in Section 4, and how to determine the level of details in perception-based motion blur rendering in Section 5. Our experimental results and discussion are given in Section 6, and finally conclusions in Section 7.

2. Related work

Motion blur rendering. Efficient rendering of motion blur effects has been a long-standing problem in interactive graphics. Recent progress can be referred in the survey paper [22]. Here, we do not cover the work on using image processing techniques for rendering blurring effect, such as [26,18,2], which is out of the scope of the paper, as our approach is aimed at using 3D models for motion blur rendering. The works with 3D models for motion blur rendering need to compute samples from the moving object for light integration, which are generally treated via rasterization or ray tracing.

The rasterization-based methods work by rasterizing the object in multidimensional spaces. These are easy to utilize graphics hardware for acceleration, such as using hardware frame buffers and GPUs. Some methods render the object in the spatial domain many times respectively and then combining the results with weights [11]. Other methods form a geometric representation for a moving primitive and rasterize the geometric representation, including oriented bounding box (OBB) in the 2D homogeneous space [1], the convex hull in the screen space [19], and temporal bounds for a tile of triangles [20].

Ray tracing has also been studied for motion blur rendering. The key computation here is to get the track for a ray from a pixel to intersect the object during a time period. With respect to this, distributed ray tracing methods have been investigated, where each ray is stochastically allocated so all dimensions are simultaneously sampled [5]. For high efficiency, different sampling techniques were proposed, such as minimum distance Poisson or jittered sampling [4], pre-computed sampling patterns [4], adaptive sampling in the Euclidean domain [10], or in the wavelet domain [23]. Recently, there are physically-based smart reconstruction methods to reason about the anisotropy of the underlying space–time imaging process [16] or samples [23,32,31], to improve the rendering quality.

Unfortunately, these rasterization-based methods and ray tracing methods did not take into account using simplified moving objects. In this paper, we mainly address this challenge to reduce the computing cost on sampling for light integration, by providing the motion-aware simplified models to the rendering methods.

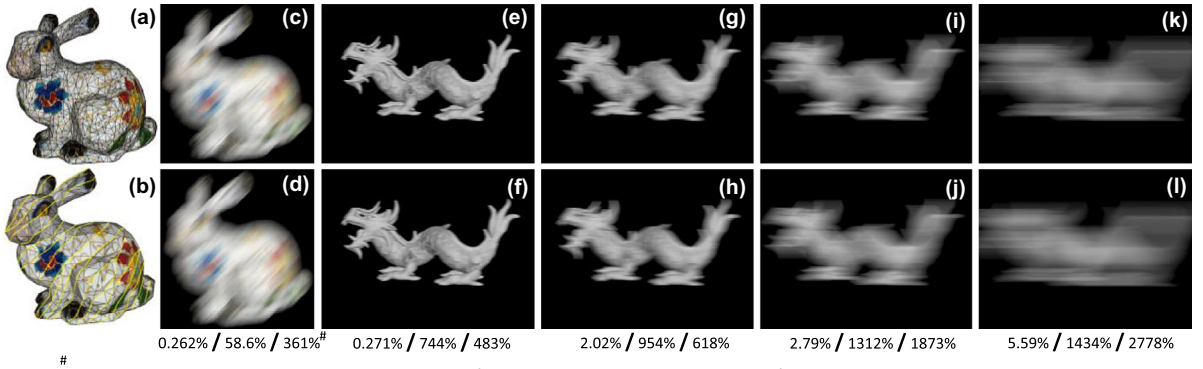

Fig. 1. Comparison of motion blur rendering results with original models and our simplified models. The percentage of computational speed-up for ray tracing, accumulation buffer and stochastic rasterization are listed at the bottom of the images. In the first row, it displays the original Bunny model and the rendering results with the original models of Bunny and Asian Dragon. In the second row, it displays our simplification manner with motion direction aware, illustrated in yellow lines here, in (b), and the rendering results with our corresponding simplified models for getting similar results as using the original models for different motions. The speedup ratios show that our method can effectively speed up existing methods for motion blur rendering, and achieve more acceleration with the larger models or faster motion. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Level of detail simplification. Level-of-detail simplification has been studied extensively for efficient rendering. In this way, geometric primitives can be used as few as possible but to render nearly similar images as using the original models, especially viewed from some distance. To this end, many simplification approaches have been investigated, such as evaluating the saliency of features for simplification [15], and efficient management of the simplified models for fast rendering [36]. A comprehensive introduction of these techniques is expounded in some survey papers [33,7]. Currently, simplification techniques basically determine the level of detail simplification by the distance from the object to the viewpoint, without considering the effect of object motion on simplification. For this reason, the features not parallel to the moving direction may be over-simplified thus impairing motion blur rendering results. Our work tries to simplify moving models for high-quality motion blur rendering as using the original models. Thus, novel algorithms are proposed with perception considered.

Perception-based rendering. There are many literatures studying psychophysical trends with the perception of computer generated stimuli, including models for the psychophysics of photo-realistic materials [29], natural illumination [14], occlusions [35], and the influence of object variety [28], in order to improve rendering efficiency. It is also studied that visual attention may affect perception, and so used for improving rendering efficiency, such as top-down visual processing [3] and quantifying the degree of blindness between images [17]. Among them, some works investigate the relationship between sensitivity of the human visual system and motion [21] and reveal that motion blur will hide detailed features, by which motion blur effects can be rendered with simplified models. However, except theoretical analysis, no technique is proposed for how to generate simplified models for motion blur rendering. In this paper, we reference the work [34] for the sensitivity variation map of the human visual system corresponding to the spatial frequency and motion of the object. Based on this work, we propose the novel perception-based approach to simplify the moving models, so that simplified models can be used for speeding up motion blur rendering in high quality.

3. Generation and perception of motion blur

Motion blur phenomenon is an integral effect of photography and film recording, which can be formalized using the following equation as given in [22].

$$I_{xy} = \int_{\Delta T} f(\omega, t) L(\omega, t) \mathrm{d}t, \tag{1}$$

where I_{xy} represents the contents of the image plane when the scene is seen in the direction ω from (x, y). Captured

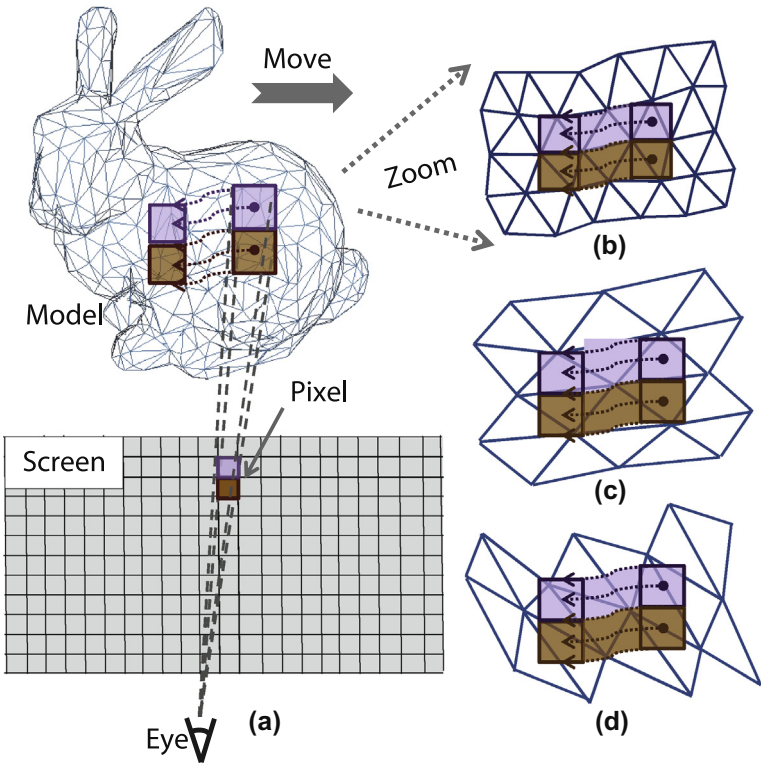

Fig. 2. Generation of motion blur effects. Content of a pixel results from the light integration over the swept trajectory on the object (a), which are enlarged to show in (b). Our direction-aware simplification simplifies triangles along the trajectories, shown in (c). For the simplification without direction aware triangles in different stripes may be processed together for simplification, shown in (d). With our simplification, it facilitates to preserve the characteristics of the features not parallel to the moving direction, to render highquality motion blur effects.

light is the result of the integration of the incoming radiance L during the exposition time when the shutter is open ΔT. $f(\omega, t)$ models the influence of optics, shutter, aperture and film. That is to say, motion blur phenomenon results from the light integration over the swept trajectory on the object, as shown in Fig. 2(a) and (b).

When a simplified model is used, the rendering result is the integration of incoming radiance produced by the new model.

$$\widetilde{I_{xy}} = \int_{\Delta T} f(\omega,t)\widetilde{L}(\omega,t)\mathrm{d}t. \tag{2}$$

with the difference caused by the simplified model expressed as

$$E_{xy} = I_{xy} - \widetilde{I_{xy}} = \int_{\Delta T} f(\omega,t)(L(\omega,t) - \widetilde{L}(\omega,t))\mathrm{d}t \tag{3}$$

By carefully simplifying the model, the radiance of simplified facet $\widetilde{L}(\omega,t)$ may be very close to $L(\omega, t)$. Moreover, according to the nature of model simplification, the normal of a generated facet for simplification is close to the mean value of the normals of its corresponding original facets, so that the radiance integration along a track on the model would have very small difference between using the original model and using the simplified model. Thus the radiance difference of each pixel E_{xy} tends to be very small. This fact makes it possible to use a simplified model to render motion blur images without producing perceivable visual difference. More details about our model simplification approach are described in Sections 4 and 5.

Actually, some works investigate the perception of moving objects, the sensitivity of the HVS (human visual system) with motion and the spatial frequency content, and attest that it is feasible to use simplified models to speed up motion blur rendering. Kelly [13] has studied this effect by measuring the threshold contrast for viewing traveling sine waves at various frequencies, where the threshold contrast is the minimum contrast at which people can distinguish the grating from the background. The Contrast Sensitivity Function (CSF) is the inverse of this measured threshold contrast, and a measure of the sensitivity of the HVS towards traveling spatial frequency patterns (more details are described in Section 5). Motivated by this work, H. Yee, et al. [34] constructed a spatiotemporal error tolerance map from psychophysical data based on velocity dependent contrast sensitivity, and applied this to motion-blurred still images, as well as animation. Moreover, F. Navarro, et al. [21] designed a series of psychophysical experiments to determine how the HVS reacts to motion blur in imagery stimuli. These works show that people are less sensitive to errors in motion blurred regions.

4. Direction-aware simplification

The radiance difference of each pixel E_{xy} is determined by the radiance of simplified model $\widetilde{L}(\omega,t)$, which is related to the normal of the simplified facets. Obviously, simplifying facets along the 'swept trajectories' will produce smaller difference, for such a processing does not treat the facets from different trajectories to generate a simplified facet, as shown in Fig. 2(c). Moreover, for the neighboring features on different trajectories in parallel with each other, their related radiances are not integrated in the rendering process. Therefore, these features can have their characteristics retained to be rendered evidently, which the human being's visual system is sensitive to. As a result, for motion blur rendering, the object should be simplified with respect to the motion direction. In other words, the features with their variation parallel to the moving direction can be simplified as much as possible, while the features with their variation vertical to the moving direction should be preserved mostly for high-quality blur rendering.

4.1. Direction-aware anisotropic simplification

Currently, simplification is always performed in an isotropic manner, and no simplification is proposed with direction-aware. Therefore, to well render motion blur effects, we design a novel simplification algorithm with direction aware. It is based on the edge contraction method [8], which is simple to implement, fast and can work locally for the parts of an object adaptively by their related level of detail simplification. In the method [8], every edge is assigned a cost to measure the shape variation if the edge is contracted. Then the edges causing less variation are decimated earlier. In our direction-aware simplification, we add a weight to the cost of every edge to earlier contract the edges that are more vertical to the moving direction, because the features with such edges are more possible to have their variations parallel to the moving direction, as shown in Fig. 2(b). Considering the weight function should be monotonically decreasing when the angle between the edge and the moving direction changes from 0 to 90 degree, we design the weight function as below, inspired by the specular term in the Phong shading model,

$$f(\alpha) = 1 - (1 - cos(\alpha))^r + \varepsilon, \tag{4}$$

where α is the angle between the edge and the moving direction, ε is a constant as a relax term to avoid giving the edges vertical to the moving direction the weight 0.0, to have them be possibly contracted for smooth simplification, and r is an index for adjusting the effect of weights. In our tests, we set ε 0.5, and get very good results. To get a suitable r in Function (4) for quality motion blur rendering, we made perception tests by assigning r many possible values, and then taking a user study. This is described in details in Appendix A. From the tests, it is known that $r = 4$ is a good choice for most models and motion speeds. And such a setting was taken for all the rendering using our approach in this paper.

5. Level of detail simplification determination

To get visually equivalent results for motion blur rendering with simplified models in the fewest facets, we should know which features are visible on a moving object. In other words, we should know the sensitivity of the human visual system to the features in viewing the moving

object. The work [13] has studied the effect by measuring threshold contrast for viewing traveling sine waves, which used the velocity of the target stimulus with respect to the retina and derived a contrast sensitivity function (*CSF*) for the sensitivity measurements. Later the contrast sensitivity function by taking into account smooth pursuit eye movements and a minimum eye velocity is given as,

$$CSF(\rho, v_R) = k \cdot c_0 \cdot c_2 \cdot v_R \cdot (2\pi\rho c_1)^2 \cdot exp(-(4\pi c_1 \rho)/\rho_{Max}) \quad (5)$$

with

$$k = 6.1 + 7.3|log(c_2 \cdot v_R/3)|^3 \quad (6)$$

and

$$\rho_{Max} = 45.9/(c_2 \cdot v_R + 2) \quad (7)$$

where v_R is the retinal velocity, measured in degrees per second (*deg/s* in short), ρ is the spatial frequency of a feature, measured in cycles per degree (*c/deg* in short), and the three coefficients $c_0 = 1.14$, $c_1 = 0.67$ and $c_2 = 1.7$. The scaling factor k is to adjust the vertical shift of sensitivity and dependent on the velocity, and ρ_{Max} is for adjusting the horizontal shift by the peak sensitivity. Both of them are for measuring the influence of the spatial and temporal components at higher frequencies.

By the work in [34], the curves by function (5) with different retinal velocities are plotted on the left in Fig. 3, where every curve corresponds to a velocity. Through the curves, we know the frequencies of features that can be perceptible under a retinal velocity. For example, according to the curve corresponding to retinal velocity $v_R = 3$ (*deg/s*), the features with their spatial frequencies below i_2 can be perceptible, though they have different sensitivities. As a result, the features with their frequencies above i_2 cannot be perceptible under such a retinal velocity, and so they can be removed in simplification. Therefore, according to function (5), we deduce the relationship between the velocity and the upper limit frequency of features that should be preserved with respect to a retinal velocity. As illustrated on the right in Fig. 3, curves are plotted for representing such relationship with different sensitivities.

5.1. Attributes

Screen velocity. Based on the above analysis, we can determine the level of detail simplification by the moving speed of an object. Here, the retinal velocity is transformed to the velocity for the object to move on the screen (the image plane), called as the *screen velocity*, as illustrated in Fig. 4. To calculate a screen velocity, we need to project the object in motion onto the screen continually with time, and record how many pixels are swept by a point of the object during a time period. Here, we represent the screen velocity in the number of pixels swept by a point of the object during a camera's shutter, namely *pixels/per shutter*. Then, by the obtained velocity for a part of the object, we know the level of detail simplification to simplify the part. In practice, models are in complex motions, and so we often handle these by decomposing complex motions into basic motions for easy treatment (More details are shown in Appendix B).

Spatial frequency. In general, the object is approximated with triangles. The size of the triangles for approximating a feature is always related to the spatial frequency of the feature. When the feature is at a higher frequency, its approximated triangles are often smaller. Similarly, when the triangles are smaller, their related features are at a higher spatial frequency in general. We approximate the spatial frequency of a triangle by its projection on the image plane. Here, we use the perimeter of the projected triangle, unlike the general way to use the area of the projected triangle, because the feature variation along the one-dimensional trajectory, corresponding to object motion, is more important for motion blur rendering. We get the number of the pixels on the perimeter, and take its reciprocal as the spatial frequency of the triangle.

5.2. Formula for computing the level of details

To get the spatial upper limit frequency for a velocity, we derive an approximation formula by Function (5), which is like the form of Function (7), because the limit perceptible spatial frequency should have some relation

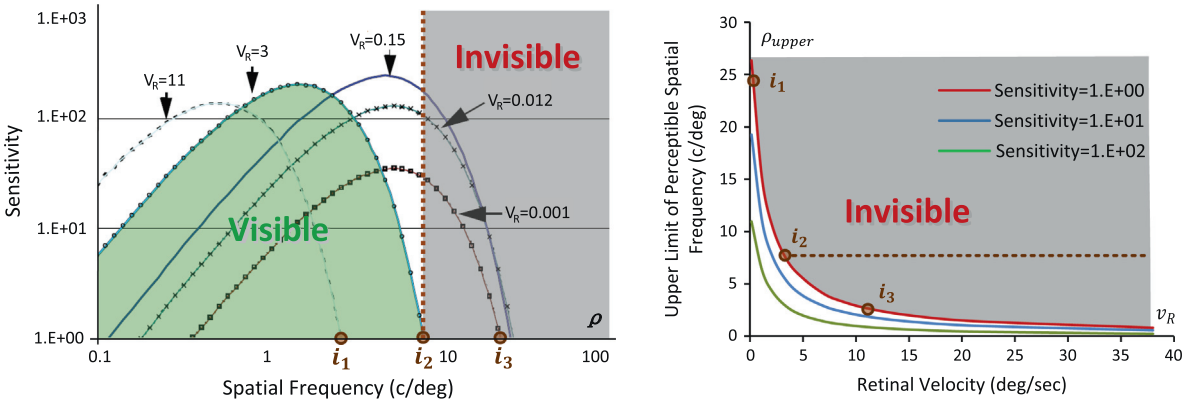

Fig. 3. Left: the relationship between the retinal velocity and the upper limit frequency, plotted by function (5) with different sensitivities. Right: the relationship between the retinal velocity and the upper limit frequency, plotted by function (5) with different sensitivities.

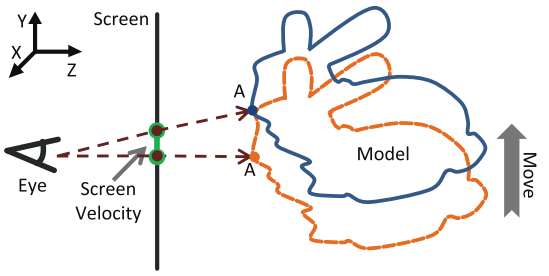

Fig. 4. In motion blur rendering, the screen can be regarded as the retina. We can simulate A's screen velocity by its projection points on the screen during a time period, where the point A moves vertically.

with the spatial frequency that causes the highest sensitivity for a velocity. The derived formula is as followed,

$$\rho_{feature} = 1.0/(\sigma_1 \cdot v_S + \sigma_2) \tag{8}$$

where v_S is our *screen velocity*, $\rho_{feature}$ is the spatial frequency of a triangle on the model, σ_1 and σ_2 are the parameters needed to be specified.

To get suitable settings of σ_1 and σ_2 for quality motion blur rendering, we conducted a user study, as presented in Appendix A. Using the weight function in Section 4, we generated the simplified models for motion blur rendering at many motion speeds, as illustrated in Fig. 5(a). Human ratings were obtained by asking viewers in the same way as in Appendix A. As a result, for a tested model at a motion speed, its simplified model with the human rating being over 0.8 can be obtained, as marked by the red line in Fig. 5(b). By collecting the curves for the tested models, we approximate a curve to represent these curves by data fitting, as illustrated in Fig. 5(c). With the approximated curve, we derive that σ_1 is 7.3 and σ_2 is 0.36.

6. Results and discussion

We have developed our model simplification method incorporated with three popular motion blur rendering approaches, to test the effectiveness of using our approach to speed up motion blur rendering. The incorporated methods are introduced in the following.

- Ray tracing based PBRT [25]. It has a geometric representation constructed for bounding the motion of the object during the time period, from the camera shutter open to close. Each ray from the camera is shot to intersect with the geometric representation, and the rays shot from a pixel have their colors averaged to give the pixel the final color. Here we mainly tested our approach to speed up the PBRT, without considering its combination with other acceleration measures.
- The rasterization-based method via the accumulation buffer [11]. It renders the moving object at many time points continuously, and averages the rendered images for motion blur rendering.
- The hierarchical stochastic rasterization method [20]. Recently, stochastic sampling has been studied for improving motion blur rendering. Such a method handles both spatial and temporal influences on distributing samples, and uses hierarchical structures for getting high efficiency.

In our tests, we downloaded the programs for PBRT [25] to run on the CPU, and implemented the methods with the accumulation buffer [11] or via hierarchical stochastic rasterization [20] in OpenGL Shading Language to run on the GPU. The tests were performed on a personnel computer installed with an Intel Core 2 Duo E8400 CPU, 4 GB RAM and an NVIDIA GeForce 8800GT GPU with 512 MB RAM. To test the acceleration efficiency with the results rendered in high quality, we shot 64 rays from every pixel using the PBRT, sampled 32 points for every pixel in implementing the method via hierarchical stochastic sampling, and the method with the accumulation buffer. Four screen velocities were used in our tests, which were 10 pixels/per shutter (Speed-1), 30 pixels/per shutter (Speed-2), 90 pixels/per shutter (Speed-3), and 270 pixels/per shutter (Speed-4) respectively. To avoid the strobing effects due to high moving speeds, we increased the sampling rates correspondingly, where 16, 48, 144 and 432 samples were used for a pixel with the four screen velocities respectively.

Quality. We first tested the rendering quality with our simplified objects, in comparison with the rendering results with the original models and the models simplified using the method in [8], based on which we developed

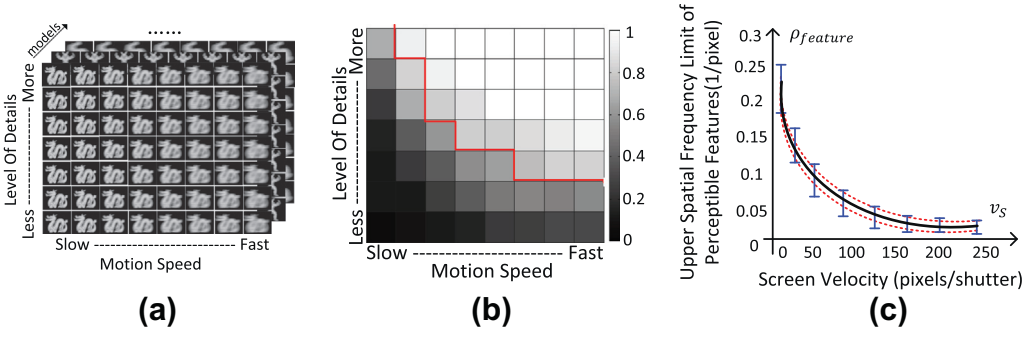

Fig. 5. (a) Motion blur rendered images by the simplified models at levels of details for motion speeds. (b) Human ratings for the motion blur rendered images in (a). (c) Our approximated curve for the red lines in (b) for the tested models, which refer to the rendered images with the human ratings just over 0.8, meaning that they can be reasonably regarded visually equivalent with the ground truth. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

our perception-based simplification. We performed a user study to evaluate our technique. The experimental environment is similar as in Appendix A. Rather than the 16 training models, we used another 5 models. In Fig. 6, it is displayed the results with some different motions, which were produced by the accumulation buffer, and the rendered images are all in 800×600 pixels. As for the rendered images by the other two methods, they are not displayed in this paper, as the images produced are the same for each case. Clearly, our results are visually equivalent with those rendered with the original models, and better than using the simplified models without motion-aware [8], though they are in same numbers of triangles for every level-of-details respectively.

We also conducted a user study for these images to know whether the viewer can distinguish the images rendered with the original models from the images rendered with the simplified models. 20 viewers took part in the study. On average, for the images rendered with our simplified models, 18/20 of them cannot distinguish, while for the images with the simplified models by [8], only 6/20 of them cannot distinguish. This means that our approach is effective for simplifying 3D models for motion blur rendering.

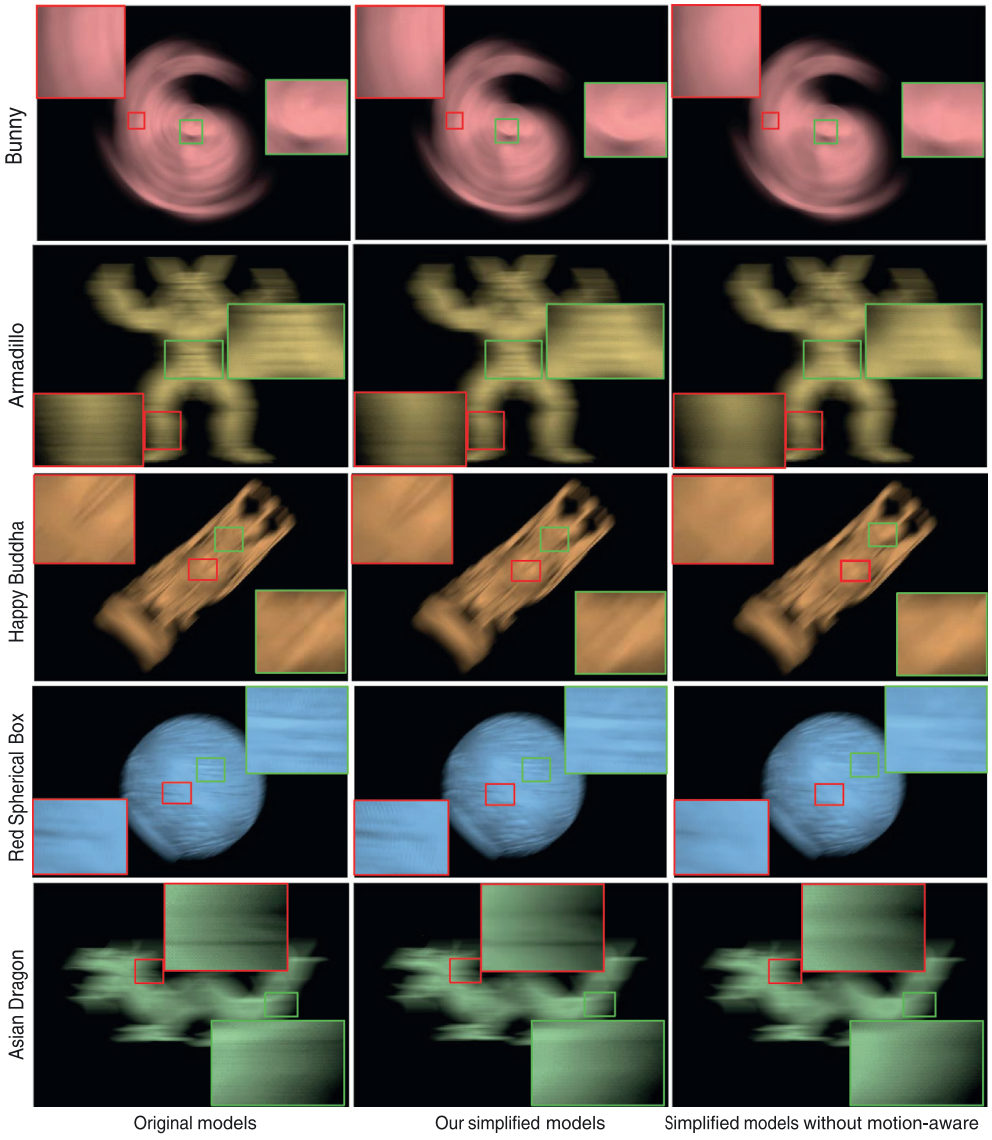

Fig. 6. The rendering quality with our simplified models is very comparative to using the original models, and better than using the simplified models without motion-aware. The features not parallel to the moving direction are well displayed in our rendering results. Left: using the original models; middle: using our simplified models; right: using the simplified models without motion-aware. The motions for the tested models here are described as follows: the Bunny model rotates at Speed-3, the Armadillo model translates horizontally at Speed-2, the Happy Buddha model translates in a non-x, y direction at Speed-2, the Red Spherical Box model moves from lower left to upper right with a rotation by itself at Speed-1, and the Asian Dragon translates horizontally at Speed-3. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 1
The tested models and the statistics data of them and their simplifications.

		Original	Simplified models with different speeds			
			Speed-1	Speed-2	Speed-3	Speed-4
Bunny	Faces	69,460	27,789	13,894	6946	2777
	Time (s)[a]	–	0.458	0.146	0.065	0.036
Armadillo	Faces	345,944	114,160	34,594	11,518	4324
	Time (s)[a]	–	3.317	1.091	0.370	0.133
Buddha	Faces	1,085,634	355,147	83,150	17,088	5913
	Time (s)[a]	–	10.95	4.077	0.978	0.167
Box	Faces	1,402,640	440,660	225,330	140,264	14,026
	Time (s)[a]	–	13.521	3.377	2.025	5.589
Asian Dragon	Faces	3,609,521	88,035	16,405	7218	3138
	Time (s)[a]	–	63.219	10.389	0.591	0.012

[a] *Time*: The time for simplifying the original models to the simplified models for Speed-1, and following the time to the next simplified model for the next higher speed, all measured in seconds.

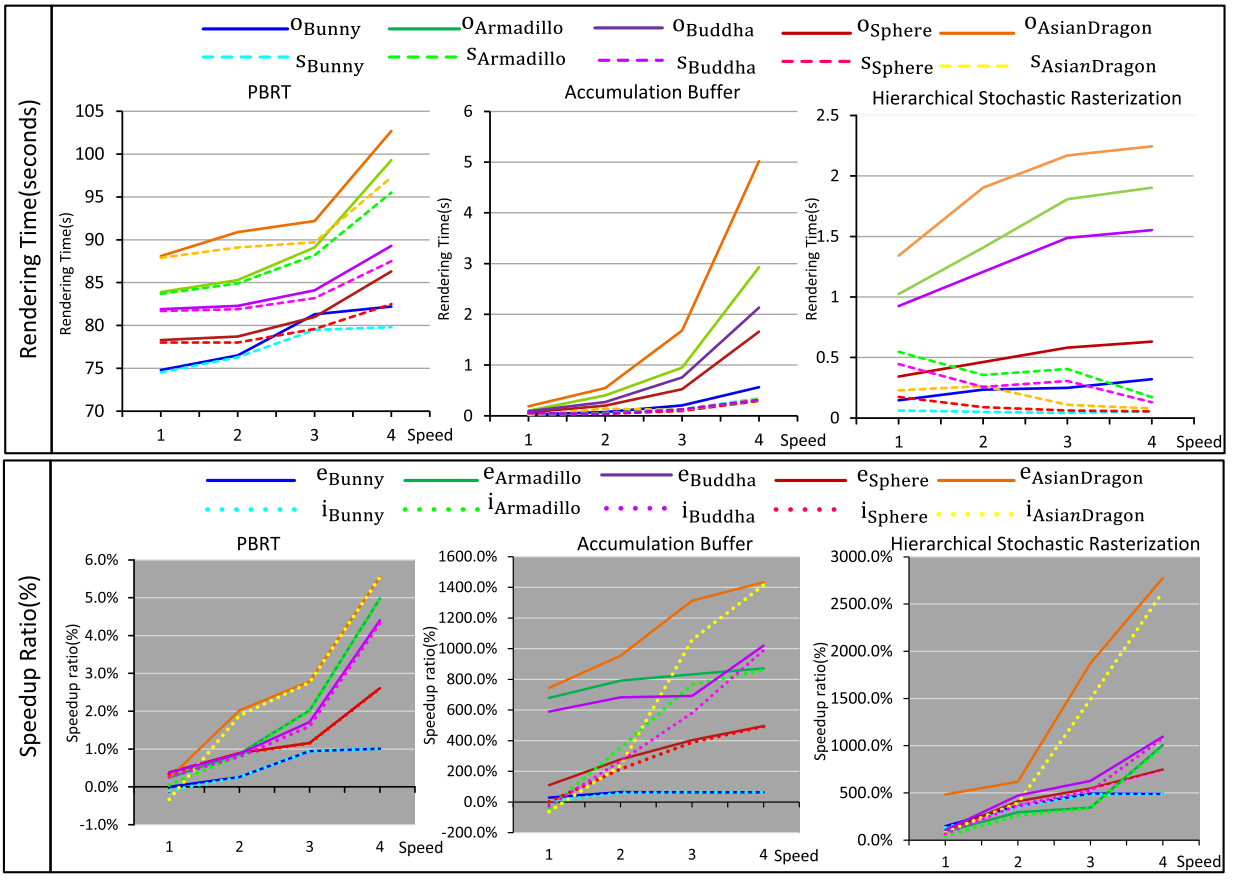

Fig. 7. The statistics of using our approach to speed up motion blur rendering. For the rendering time, the model names with a prefix 'o' or 's' refer to using the original models or our simplified models. For the speedup ratios, the model names with a prefix 'e' or 'i' refer to the speedup ratios computed with the time on simplification excluded or included.

Speed. In Table 1, we show the tested models and the statistics data of them and their simplifications. With the higher moving speed, the models can be simplified more, and more acceleration can be achieved for rendering, as shown in Fig. 7. If the model is simplified more, the time spent on simplification would be more. In practice, when the motion speed changes gradually, the model is simplified or refined gradually, so that the processed models for a speed can be used to produce the next one for a higher or lower speed [12]. Thus, the time on simplification can be reduced much, in comparison with simplifying the model from the original one overhead for any given moving speed, as shown in Table 1. This is useful to deal with the cases when the object changes the speed frequently in motion.

In our motion-blur rendering tests, the acceleration ratio is computed by $SpeedupRatio = (Time_{orig}\text{-}Time_{our})/Time_{our}$, where $Time_{orig}$ and $Time_{our}$ refer to the time for rendering 100 frames with the original model and our simplified model. From the statistics in Fig. 7, we know that our simplified models can efficiently speed up motion blur rendering, and with the moving speed higher, the acceleration ratios also increases. Comparatively speaking, our method speeds up the rasterization-based method more significantly. This is due to the fact that the rasterization-based rendering method is in a time complexity $O(n)$, while the ray tracing based methods in a time complexity $O(logn)$ due to acceleration structures for ray-object intersection, where n is the number of the facets of a model. The statistics shows that the rasterization-based methods can be accelerated largely, even over 27 times. Our approach is effective to speed up the rendering of the models with a large number of facets. For instance, the speedup ratios for the Asian dragon model are much higher than that for the Bunny model. This is because, for a fixed image size, the perceptible features of any model are in similar sizes so that the larger model has more facets simplified. Though simplification needs some time, which may slow down the rendering speed, the time is not long and can be amortized in rendering the related frames. However, when the moving speed is low, small-sized features are then perceptible so that models cannot be simplified significantly. In such a case, due to the time cost on simplification, the rendering speed may even decrease. Therefore, when the object moves slowly, it is not suitable to apply our method. In our tests, when the velocity is below 30 pixels/per shutter, our method may slow down motion blur rendering, especially for the large model as the time cost on simplification may take up a large proportion of the rendering time.

Textured objects. Our approach also supports motion blur rendering of textured objects, as shown in Fig. 1(d). Though there are techniques for simplifying textured objects, we only adopt a simple measure to treat texture coordinates in simplification. This can get high quality rendering results because motion blur hides detailed features. By this technique, when some edges are collapsed for simplification, the generated vertices have their texture coordinates obtained by averaging the texture coordinates of their respectively replaced vertices.

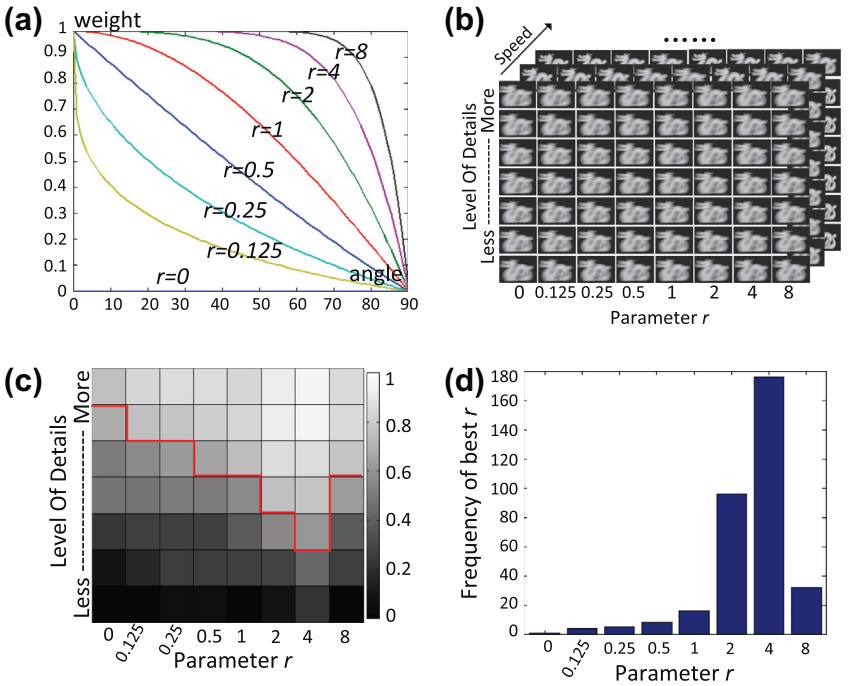

Fig. 8. (a) Weight function curves with different settings of parameter r. (b) Motion blur rendered images by the simplified models with different settings of r, and related to different motion speeds. (c) Human ratings for the motion blur rendered images in (b). (d) The statistical histogram for the r settings to have higher human ratings.

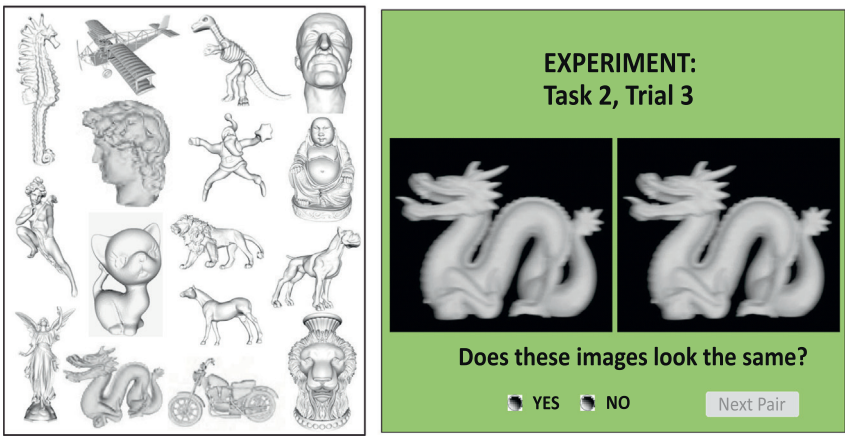

Fig. 9. Left: The 16 selected models used in our tests. Right: The user interface used in our tests. Both the ground truth and our results with simplified models are presented.

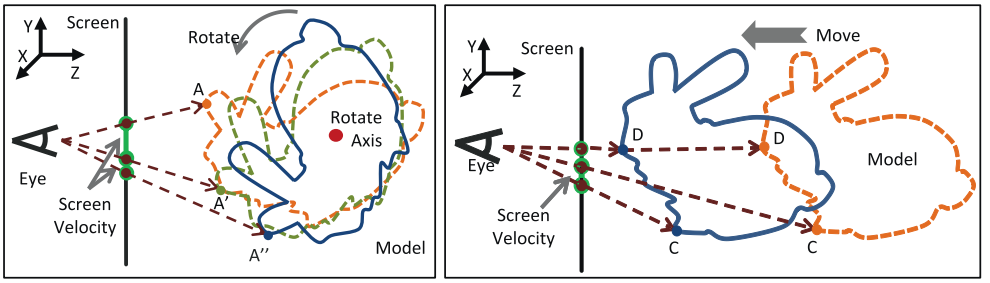

Fig. 10. Two typical complex motions. Left: the object rotating around an axis not parallel to the *z*-axis. Right: the object translating along a direction parallel to the *z*-axis.

7. Conclusions

In this paper, by tackling the impact of motion on simplification, we develop novel algorithms to adaptively simplify models in motions, and deduce a formula to generate the level of detail simplification related to the moving speed of the object, motivated from the psychophysical studies. Therefore, simplified models can be effectively used to speed up the existing methods for motion blur rendering with 3D models. Our experimental results have shown that using our perception-based approach, motion blur rendering can be sped up significantly, and the speed-up ratios are even higher when the object is larger or moves faster.

Our approach has attested that models can be simplified to promote motion blur rendering, but there are further work for related applications. For example, our current system takes somehow higher time cost on simplification, which may make it difficult to generate usable simplified models in time when the object changes motion dramatically. It is necessary to further study how to speed up our approach, including using parallel processing techniques. Another interesting issue is to extend our approach to deal with deformable objects, which are widely used in games and animation.

Acknowledgments

The work is partially supported by the National Natural Science Foundation of China (No. 61379087), the Knowledge Innovation Program of the Chinese Academy of Sciences, and RGC research grant (Ref. 416212), UGC direct grant for research (Nos. 2050485, 2050454).

Appendix A. Perception tests

Our perception tests were carried out using a 22in TFT-LCD monitor with a resolution of 86.27dpi (1680×1050) and a luminance of 200 (cd/m^2). The viewing distance from the observers' eyes to the monitor was 0.5 m. This is typical of the visual environment for desktop applications. In our rendering, static lights were used with a black background, and the FOV was 33.33 degrees. The images were generated by the accumulation buffer method using OpenGL, for it is fast and can reveal the details very well. These generated results are similar with those generated by ray-tracing methods and stochastic sampling methods, so that we did not use ray-tracing methods and stochastic sampling methods in the perception tests.

To search for a good r value in Function (4), we assign r 8 values to test, which are 0.125, 0.25, 0.5, 0, 1, 2, 4 and 8, and their corresponding weight functions with the angle α are plotted in Fig. 8(a). It should be noted that in this plot, the weight function with $r = 0$ is a horizontal line with a constant ε, which leads the model simplified without direction aware.

For a given r value, we generated many simplified models at different levels of details with the related weight function. Then, for any a simplified model, we took it for motion blur rendering by motion speeds. Thus, we collected motion blur images for an object, as shown in Fig. 8(b). Afterwards, we used the original model for motion blur rendering by the motion speeds for rendering the simplified models, and took the scheme of force two-alternative choice (FAC) to ask viewers whether the rendered images with the simplified models are looked as the rendered image with the original model, for a speed. By the statistics, we computed the human rating for a rendered image with a simplified model, computed as $u_1/(u_1 + u_2)$, where u_1 is the number of the viewers who regarded it as the same with the rendered image by the original model, and u_2 is the number of the viewers who did not. Such ratings are illustrated in Fig. 8(c).

By the human ratings, it is reasonable to regard the images with its rating value over 0.8 are visually equivalent with the ground truth. By collecting such images, we can get a statistical histogram for r, as illustrated in Fig. 8(d), which is obtained by inviting 20 viewers and using 16 models. The 16 models are displayed in the left in Fig. 9, and the user interface for FAC is shown in the right in Fig. 9.

Appendix B. Complex motions

In applications, motion blur can be produced by motions of both objects and the camera. But all these can be transformed to have the camera fixed and objects move in axis-aligned directions for rendering the motion blur effects, by which we discuss our approach without loss of generality in this paper. For a model with complex motion, we can simplify its parts adaptively by their corresponding screen velocities at every time point in principle. But, frequent simplification operations are not beneficial for motion blur rendering, as they may cause much computing cost. Thus, we conservatively generate the simplified object for motion blur rendering during a time period, promising to render all the frames in high quality during the time period no matter how the object moves.In other words, by the motion speeds and motion manners in a time period, we produce only a simplified model conservatively, and use it for motion blur rendering in this time period. As for the measures to conservatively get a simplified model for complex motion, they are discussed in the following paragraphs.

Motions can be basically classified into two categories: translation and rotation (as shown in Figs. 4 and 10). Object translation can be classified as along the x, y and z directions respectively, in the right-hand coordinate system with the xy plane as the screen plane and the z direction pointed from left to right in this paper. For a translation along the x or y direction (as shown in Fig. 4), we can project the object perspectively onto the screen, and compute the screen velocities for the parts of the object respectively. When an object rotates around a direction parallel to the z-direction, any part of it will not have the z-coordinates changed. So the object can be simplified as for its translation along the x or y axis, except that the projected trajectory to be tracked here is an arc rather than a straight line as for a translation.

In complex cases, the object may rotate around a direction that is not parallel to the z-direction, or the object translates along a direction parallel to the z axis, as shown in Fig. 10. For the first case, a part of the object may have its z-coordinate changed during the rotation. This can cause the screen velocity changing quickly, as illustrated by the point A. Therefore, velocity of point A would be ever-changing and hard to compute. For the sake of both simplicity and conservation, we determine the screen velocity by its nearest position to the viewer, because this is the lowest screen velocity and will evoke simplification to the least. This is aimed to avoid complex computation to save time.

In the second case when an object moves along a direction parallel to the z axis (as shown on the right in Fig. 10), the screen velocity for the object is caused by foreshortening. For the part farthest from the viewing direction, like C, it has the highest screen velocity. As for the point D, the intersection point between the viewing direction and the object, it would have 'zero' screen velocity because its projection point on the screen is not changed. Like the conservative treatment in the last paragraph when the screen velocity of a part changes quickly, we will not take into account the influence of the translation along the direction parallel to the z axis in practice.

References

[1] T. Akenine-Möller, J. Munkberg, J. Hasselgren, Stochastic rasterization using time-continuous triangles, in: Proceedings of the 22nd ACM SIGGRAPH/EUROGRAPHICS Symposium on Graphics Hardware, GH'07, 2007, pp. 7–16.
[2] G.J. Brostow, I. Essa, Image-based motion blur for stop motion animation, in: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '01, ACM, New York, NY, USA, 2001, pp. 561–566.
[3] K. Cater, A. Chalmers, P. Ledda, Selective quality rendering by exploiting human inattentional blindness: looking but not seeing, in: Proceedings of the ACM Symposium on Virtual Reality Software and Technology, VRST '02, ACM, New York, NY, USA, 2002, pp. 17–24.
[4] R.L. Cook, Stochastic sampling in computer graphics, ACM Trans. Graph. 5 (1) (1986) 51–72.
[5] R.L. Cook, T. Porter, L. Carpenter, Distributed ray tracing, SIG-GRAPH Comput. Graph. 18 (3) (1984) 137–145.
[6] K. Egan, Y.-T. Tseng, N. Holzschuch, F. Durand, R. Ramamoorthi, Frequency analysis and sheared reconstruction for rendering motion blur, ACM Trans. Graph. 28 (3) (2009) 93:1–93:13.
[7] M. Garland, Multiresolution modeling: survey & future opportunities, in: Eurographics '99 – State of the Art Reports, 1999, pp. 11–131.
[8] M. Garland, P.S. Heckbert, Surface simplification using quadric error metrics, in: Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '97, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 1997, pp. 209–216.
[9] A.S. Glassner, Spacetime ray tracing for animation, IEEE Comput. Graph. Appl. 8 (2) (1988) 60–70.

[10] T. Hachisuka, W. Jarosz, R.P. Weistroffer, K. Dale, G. Humphreys, M. Zwicker, H.W. Jensen, Multidimensional adaptive sampling and reconstruction for ray tracing, ACM Trans. Graph. 27 (3) (2008) 33:1–33:10.
[11] P. Haeberli, K. Akeley, The accumulation buffer: hardware support for high-quality rendering, SIGGRAPH Comput. Graph. 24 (4) (1990) 309–318.
[12] H. Hoppe, Progressive meshes, in: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '96, ACM, New York, NY, USA, 1996, pp. 99–108.
[13] D.H. Kelly, Motion and vision. ii. Stabilized spatio-temporal threshold surface, J. Opt. Soc. Am. 69 (10) (1979) 1340–1349.
[14] J. Křivánek, M. Fajardo, P.H. Christensen, E. Tabellion, M. Bunnell, D. Larsson, A. Kaplanyan, Global illumination across industries, in: ACM SIGGRAPH 2010 Courses, SIGGRAPH '10, ACM, New York, NY, USA, 2010.
[15] C.H. Lee, A. Varshney, D.W. Jacobs, Mesh saliency, ACM Trans. Graph. 24 (3) (2005) 659–666.
[16] J. Lehtinen, T. Aila, J. Chen, S. Laine, F. Durand, Temporal light field reconstruction for rendering distribution effects, ACM Trans. Graph. 30 (4) (2011) 55:1–55:12.
[17] L.-Q. Ma, K. Xu, T.-T. Wong, B.-Y. Jiang, S.-M. Hu, Change blindness images, IEEE Trans. Vis. Comput. Graph. 19 (11) (2013) 1808–1819.
[18] N.L. Max, D.M. Lerner, A two-and-a-half-d motion-blur algorithm, in: Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '85, ACM, New York, NY, USA, 1985, pp. 85–93.
[19] M. McGuire, E. Enderton, P. Shirley, D. Luebke, Real-time stochastic rasterization on conventional gpu architectures, in: Proceedings of the Conference on High Performance Graphics, HPG '10, Eurographics Association, 2010, pp. 173–182.
[20] J. Munkberg, P. Clarberg, J. Hasselgren, R. Toth, M. Sugihara, T. Akenine-Möller, Hierarchical stochastic motion blur rasterization, in: Proceedings of the ACM SIGGRAPH Symposium on High Performance Graphics, HPG '11, ACM, New York, NY, USA, 2011, pp. 107–118.
[21] F. Navarro, S. Castillo, F.J. Sero' n, D. Gutierrez, Perceptual considerations for motion blur rendering, ACM Trans. Appl. Percept. 8 (3) (2011) 20:1–20:15.
[22] F. Navarro, F.J. Sern, D. Gutierrez, Motion blur rendering: state of the art, Comput. Graph. Forum 30 (1) (2011) 3–26.
[23] R.S. Overbeck, C. Donner, R. Ramamoorthi, Adaptive wavelet rendering, ACM Trans. Graph. 28 (5) (2009) 140:1–140:12.
[24] K. Pauwels, M. Van Hulle, Realtime phase-based optical flow on the gpu, in: IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, 2008 (CVPRW '08), 2008, pp. 1–8.
[25] M. Pharr, G. Humphreys, Physically Based Rendering, second ed., From Theory To Implementation, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2010.
[26] M. Potmesil, I. Chakravarty, Modeling motion blur in computer generated images, SIGGRAPH Comput. Graph. 17 (3) (1983) 389–399.
[27] J. Ragan-Kelley, J. Lehtinen, J. Chen, M. Doggett, F. Durand, Decoupled sampling for graphics pipelines, ACM Trans. Graph. 30 (3) (2011) 17:1–17:17.
[28] G. Ramanarayanan, J. Ferwerda, B. Walter, K. Bala, Visual equivalence: towards a new standard for image fidelity, in: ACM SIGGRAPH 2007 Papers, SIGGRAPH '07, ACM, New York, NY, USA, 2007.
[29] H. Rushmeier, The perception of simulated materials, in: ACM SIGGRAPH 2008 Classes, SIGGRAPH '08, ACM, New York, NY, USA, 2008, pp. 7:1–7:12.
[30] J. Schmid, R.W. Sumner, H. Bowles, M. Gross, Programmable motion effects, ACM Trans. Graph. 29 (4) (2010) 57:1–57:9.
[31] P. Sen, S. Darabi, On filtering the noise from the random parameters in monte carlo rendering, ACM Trans. Graph. 31 (3) (2012) 18:1–18:15.
[32] P. Shirley, T. Aila, J. Cohen, E. Enderton, S. Laine, D. Luebke, M. McGuire, A local image reconstruction algorithm for stochastic rendering, in: Symposium on Interactive 3D Graphics and Games, I3D '11, ACM, New York, NY, USA, 2011, pp. 9–14.
[33] O.M. van Kaick, H. Pedrini, A comparative evaluation of metrics for fast mesh simplification, Comput. Graph. Forum 25 (2) (2006) 197–210.
[34] H. Yee, S. Pattanaik, D.P. Greenberg, Spatiotemporal sensitivity and visual attention for efficient rendering of dynamic environments, ACM Trans. Graph. 20 (1) (2001) 39–65.
[35] I. Yu, A. Cox, M.H. Kim, T. Ritschel, T. Grosch, C. Dachsbacher, J. Kautz, Perceptual influence of approximate visibility in indirect illumination, ACM Trans. Appl. Percept. 6 (4) (2009) 24:1–24:14.
[36] Q. Zhu, J. Zhao, Z. Du, Y. Zhang, Technical section: quantitative analysis of discrete 3d geometrical detail levels based on perceptual metric, Comput. Graph. 34 (1) (2010) 55–65.