www.ietdl.org

Published in IET Image Processing
Received on 27th January 2013
Revised on 16th April 2013
Accepted on 18th May 2013
doi: 10.1049/iet-ipr.2013.0067

ISSN 1751-9659

Adaptive sampling approach for volumetric shadows in dynamic scenes

Hsiang-Yu Lin[1], *Chin-Chen Chang*[2], *Yu-Ting Tsai*[3], *Der-Lor Way*[4], *Zen-Chung Shih*[1]
[1]*Institute of Multimedia Engineering, National Chiao Tung University, Hsinchu 300, Taiwan*
[2]*Department of Computer Science and Information Engineering, National United University, Miaoli 360, Taiwan*
[3]*Department of Computer Science and Engineering, Yuan Ze University, Zhongli, Taiwan*
[4]*Department of New Media Art, Taipei National University of Arts, Taipei, Taiwan*
E-mail: ccchang@nuu.edu.tw

Abstract: Ray marching is an important technique to generate volumetric lighting effects. However, it is very expensive for each pixel on the screen, especially in dynamic scenes. The authors propose an adaptive approach to reduce samples for volumetric shadows in the time domain. In dynamic scenes, shadow volumes of moving objects are created and are rasterised to decide pixels that cannot be reused. The authors use a stencil buffer to maintain the information of screen pixels by recomputing or just using the previous information. Experimental results show that the proposed approach is simple to implement. Moreover, compared to the previous method, the proposed approach achieves a specific speedup and maintains similar visual quality.

1 Introduction

Realistic rendering is important to the movie and gaming industries. However, it is a challenging task for simulating radiance transport in participating media efficiently and accurately. Some simplifying assumptions are often made about scenes to maintain interactivity. One common assumption limits light interactions to surfaces. It ignores contributions from light scattered by particles in the air. This means that radiance is constant along rays between surfaces. However, there are many real-world situations where this assumption is inaccurate, such as scattering from particles in the air, volumetric shadows and so on. Scattering in participating media generates volumetric lighting effects, which greatly enhance the realism in virtual scenes. However, the computation cost is very expensive as scattering occurs at every point in sampling space.

For single-scattering effects, ray marching [1] is usually used, which needs to integrate in-scattered light along a ray emanating from the viewpoint. Consequently, the computation cost scales with the number of rays and samples along the ray. The previous work [2] uses down sampling to reduce the number of rays, and restricts ray marching regions to lit segments. Another approach [3] uses irregular sampling and interpolates to reduce the number of rays. However, the computation cost will be very expensive for dynamic scenes.

In this paper, we present an adaptive sampling algorithm in the time domain for volumetric shadows. We can reduce the number of rays by reusing the results from the previous frame in dynamic scenes. We determine pixels on the screen whether they are needed to be updated or not. We focus on moving objects that are changed frame by frame and make a stencil buffer to mask pixels that are not reusable by rasterising these objects. The proposed adaptive sampling technique for volumetric shadows in dynamic scenes can be combined into exist rendering systems with similar visual quality and yields a definite speedup.

The contributions of the proposed approach are as follows: (a) We present an adaptive sampling algorithm for volumetric shadows in dynamic scenes; (b) The proposed method can be easily integrated to other rendering techniques since two rendering passes are only needed; (c) The proposed method gains 10–40% speedup with similar visual quality compared to the previous method.

The rest of this paper is organised as follows: First, in Section 2, we review some related works. Section 3 describes the proposed adaptive sampling algorithm for volumetric shadows in dynamic scenes. Section 4 describes the results. Finally, Section 5 gives conclusions and future works.

2 Related works

Several approaches have been proposed for rendering participating media. Ray marching [1] is a traditional method for computing volumetric shadows with single scattering. For each pixel on the screen, a ray is cast from the viewpoint and the approximated radiance is computed by sampling along the ray. If the point is occluded from the light, it is discounted from the integral. The ray terminates once it hits an object's surface.

Several approaches work on solving single-scattering integral semi-analytically [4, 5], but these studies ignore shadowing. Other methods like volumetric photon mapping [6, 7] and line space gathering [8] dedicate to more difficult scattering issues, such as volumetric caustics or multiple scattering. These problems are far from real time on complex scenes. Max [9] computed the single-scattering integral by searching the lit parts on every ray using shadow volumes intersected with epipolar slices and then analytically determined the integral on each lit part. Dobashi *et al.* [10] presented a method to compute the scattering integral by constructing slices at different depths, rendering the scattering at these slices, and incorporating them. Imagire *et al.* [11] proposed an approach that combines both slices and ray marching.

Since previous approaches for volumetric shadows [12, 13] have to consider about shadow boundaries, it takes additional work. Wyman and Ramsey [2] used shadow volumes [14, 15] to distinguish the shadow regions of each ray. This ray marching technique can reach a definite speedup compared to the traditional ray marching. However, ray marching with participating media still consumes a lot of performance, especially in dynamic scenes. Hu *et al.* [16] applied Wyman and Ramsey's method [2] in their algorithm for interactive volumetric caustics. It works for simple occlusion, but it is slow when visibility boundaries become complex. The algorithm of Billeter *et al.* [17] generates the shadow volume extruded from a shadow map and computes the lit segments with a graphical processing units (GPU) rasteriser. It performs well for low-resolution shadow maps, and the performance is dependent on shadow map resolution.

Volumetric shadow problems can be simplified using epipolar geometry [3, 9]. Max [9] updated the hidden segments of view rays along epipolar planes and then computed scattering integrals of lit segments analytically. The complexity of the visibility function is proportional to integration cost. This shadow volume algorithm is dependent on scene complexity. Recently, Engelhardt and Dachsbacher [3] noticed that the values of the scattering integral vary smoothly along epipolar plane lines in most cases except at depth discontinuities. They sampled more at discontinuities which occurred due to occlusion in image space. A Z-buffer is used for detecting, and the visibility is queried from the shadow map. Compared to ray marching, this sampling strategy can speed up for most cases.

The incremental integration method is proposed by Baran *et al.* [18]. They used epipolar rectification to reduce computation of scattering integral and accelerate this computation with a partial sum tree. This method is fast on the CPU, but hard to implement on GPU due to incremental traversal of camera rays in specified order. For utilising GPU parallelism, Chen *et al.* [19] used a 1D min–max data structure to avoid dependence between camera rays. They do not have to do camera rectification and avoid processing twice as many as camera rays. Their results show that their approach can speed up and slightly obtain better quality. Wyman [20] also utilised epipolar space with voxelisation. They reduced visibility query cost by voxelising the scene into a binary, epipolar space grid. A fast parallel scan is then used to identify shadowed voxels. Once identified, a texture can be built according to this voxelised shadow volumes. Then, 128 visibility samples along the eye ray can be done with a single texture fetch. Engelhardt and Dachsbacher [3] reached a definite speedup. However, there are some improvements by reusing temporal information.

3 Proposed approach

The proposed approach for computing light L at the viewpoint is based on the model [2, 21]

$$\begin{aligned} L &= L_{\text{sl}} + L_{\text{se}} \\ &= L_p \mathrm{e}^{-(k_\text{a}+k_\text{s})d_p} + \int_0^{d_p} k_\text{s}\rho(\alpha)\frac{I_0}{d^2}\mathrm{e}^{-(k_\text{a}+k_\text{s})(d+x)}\mathrm{d}x \end{aligned}$$

where two terms L_{sl} and L_{se} represent surface lighting directly reflected from a point p on an object and scattering effects in-scattered along the viewing ray, respectively. The light reflected from point p to the viewpoint is denoted as L_p; I_0 is emitted radiance of light; the distance from the viewpoint to p is d_p; and d is the distance from the light to a point on the viewing ray at distance x from the viewpoint. k_a and k_s are absorption and scattering coefficients, respectively, and $\rho(\alpha)$ is the phase function.

We first create a shadow map from the light as follows: We first render the scene from the light and save the depths as shadow map. Then, we render the scene from the viewpoint and compare the depths for a given point sp as follows

$$\begin{cases} \text{If } R = D, & \text{point } sp \text{ is not in shadow} \\ \text{If } R > D, & \text{point } sp \text{ is in shadow} \end{cases}$$

where R is the distance from the point to the light and D is the depth value in the shadow map.

Then, we render a shadow volume from the viewpoint and store the distances to the front-most and back-most polygons. For surface lighting, it is a general rendering process and can be computed quickly based on the Phong lighting model and the shadow map. For scattering effects, they can be computed by the proposed temporally adaptive sampling algorithm based the shadow map and the shadow volume which will be described in more detail in the following sections.

3.1 Temporally adaptive sampling algorithm

The main steps of the proposed adaptive sampling algorithm for volumetric shadows in dynamic scenes are described as follows:

1. Check whether the previous frame exists or not.
2. If the previous frame does not exist, compute scattering effects with the hybrid ray marching technique [2].
3. If the previous frame exists, first generate a stencil mask and determine the changed part of the previous frame, and then combine the previous frame with the changed part of the previous frame to compute scattering effects.

The stencil mask is used to classify pixels. This stencil mask is made by considering shadow volumes [14, 15] of moving objects and moving objects. Also, the current positions and the previous positions of moving objects are used to generate such a mask. Fig. 1 depicts the flowchart of the proposed temporally adaptive sampling algorithm for computing scattering effects.

3.2 Shadow volumes of moving objects

The shadow volume of moving objects is an important part to determine changed pixels of the screen since visibility of particles within the shadow volume will be changed in dynamic scenes. Therefore, the proposed algorithm generates a shadow volume of moving objects from the viewpoint. Because of participating media, samples in the scene influence the results of ray marching. It is important for deciding pixels whether they are changed or not. If visibility of samples in the scene along a viewing ray is changed, the radiance of this ray should be computed. Otherwise, the ray marching result of the previous frame is directly used.

Without loss of generality, assume that which moved objects are known. The procedure for generating a shadow

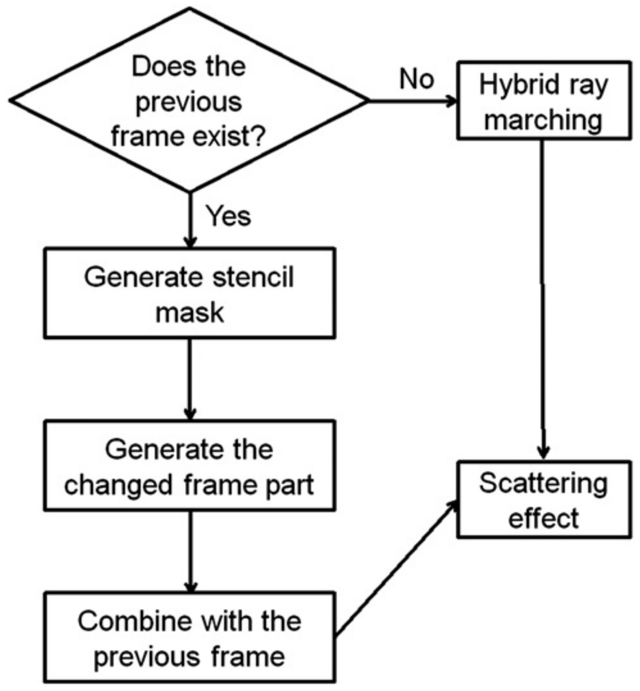
Does the previous frame exist?
No
Yes
Hybrid ray marching
Generate stencil mask
Generate the changed frame part
Combine with the previous frame
Scattering effect

Fig. 1 *Flowchart of the proposed temporally adaptive sampling approach for scattering effects*

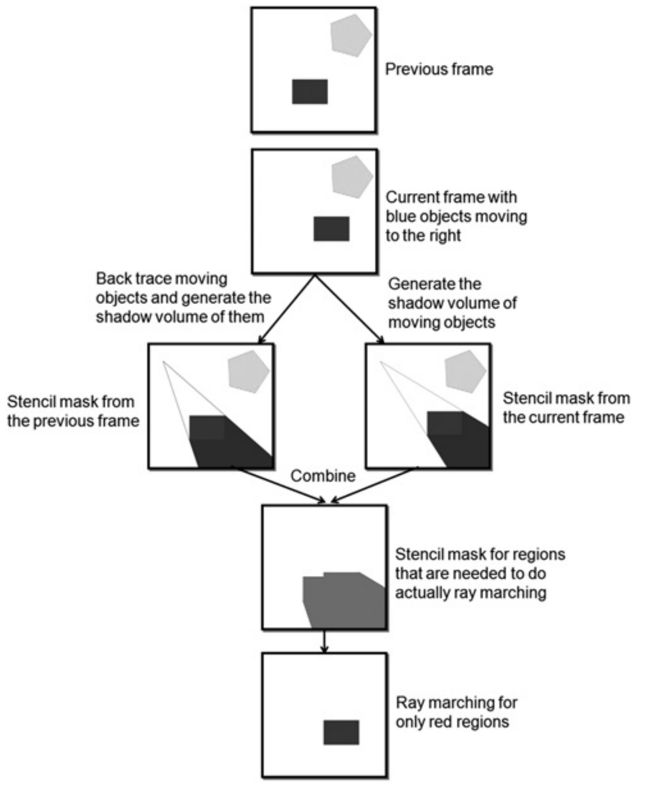
Previous frame
Current frame with blue objects moving to the right
Back trace moving objects and generate the shadow volume of them
Generate the shadow volume of moving objects
Stencil mask from the previous frame
Stencil mask from the current frame
Combine
Stencil mask for regions that are needed to do actually ray marching
Ray marching for only red regions

Fig. 2 *Generate shadow volumes of moving objects of both the previous frame and the current frame, and obtain a stencil mask from the generated shadow volumes*

volume within a shader is as follows: First, silhouettes of the moving objects are generated. Then, two more vertices for each silhouette are added to produce shadow polygons. Hence, the shadow volume is constructed.

3.3 Dynamic stencil mask

The stencil mask of current moving objects is not enough for specifying all changed pixels since it only includes samples that are changed from visible to invisible. It is also needed to consider visibility of samples that are changed from invisible to visible. This usually happens in the shadow volume of moving objects of the previous frame. Therefore, both the previous and current shadow volumes of moving objects are considered. Actually, all shadow volumes of moving objects from the previous frame to the current frame are considered.

The procedure for generating a stencil mask, shown in Fig. 2, is as follows:

1. Back trace positions of moving objects of both the previous frame and the current frame.
2. Generate shadow volumes of the moving objects of both the previous frame and the current frame.
3. Compute two stencil masks from the generated shadow volumes of both the previous frame and the current frame.
4. Combine the two stencil masks into the combined mask to determine the regions that are needed to do actually ray marching.

For calculating the stencil mask, based on the function glStencilFunc, the mask is calculated by specifying a mask mask(i, j) that is ANDed with both the reference value ref (i, j) and the stored stencil value stencil_nuffer(i, j). The formula for calculating the mask is defined as

$$\text{mask}(i,j) = \text{ref}(i,j)\ \text{AND}\ \text{stencil\_buffer}(i,j)$$

The stencil mask includes not only rasterisation of moving objects and the shadow volume, but also the parts in the previous frame. This is done with stencil operations and stencil function sets. Stencil operations specify actions when stencil or depth tests pass or not. When building this mask, the proposed approach does not need to consider about depth information, and depth test is disabled. Stencil functions specify stencil tests that decide whether pixels pass the mask or not. During the rendering pass, all the changed parts of the screen are set to non-zero in the stencil mask. After setting stencil operations and stencil functions, a shader is performed to generate a shadow volume of moving objects. This shader is performed twice separately for the current positions of moving objects and the same moving objects but with positions of the previous frame. Pixels with different ray marching results due to moving objects are then marked as non-zeroes in the stencil mask. We store different rendering passes as different textures using off-screen rendering techniques. The frame buffer object of OpenGL is convenient to do this, but it is only a manager of memory to help us control off-screen render. Each render buffer is created before it is used.

Notice that there is no need to compute all pixels in the current frame. This combined mask helps to indicate which pixels are needed for ray marching because of movements of objects. This combined mask is used to compute pixels that are marked changed. Fig. 3 indicates that the results from the previous frame and the current frame are combined by updating red regions from the previous frame.

Moreover, the types of motion of moving objects can affect the dynamic stencil mask. A linear motion will not lose pixels in general, but a rotate motion may leave some pixels out of the mask. This situation can be handled by updating whole

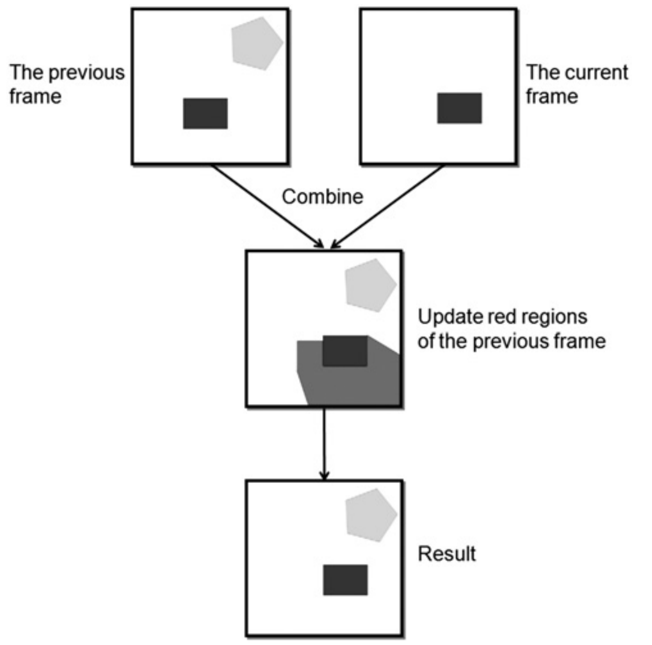

Fig. 3 *Combine results from both the previous frame and the current frame*

scene with a definite number of frames. We call this number as the reused frame number, which will be described in the following section.

3.4 Combining temporal frame

The proposed approach works on the screen space. We shot a ray for each pixel and then compute the radiance for each pixel. After building the stencil mask, a texture with results of the changed parts can be generated. Assume that the camera is fixed and it is only needed to compute the background texture once. The whole scene is first computed as the background. Then a shader is applied to combine this background texture and the current changed part of the texture. Actually, it is not necessary to process every pixel. The background texture is just taken as render targets and process only pixels that are within the stencil mask. We compute the whole scene as per the reused frame number, as shown in Fig. 4.

If the reused frame number is set to more than a definite number, some unwanted colours would be accumulated and it produces obvious artefacts. In the experiments, the reused frame number is set as 3 because the stencil mask sometimes cannot fully cover the regions.

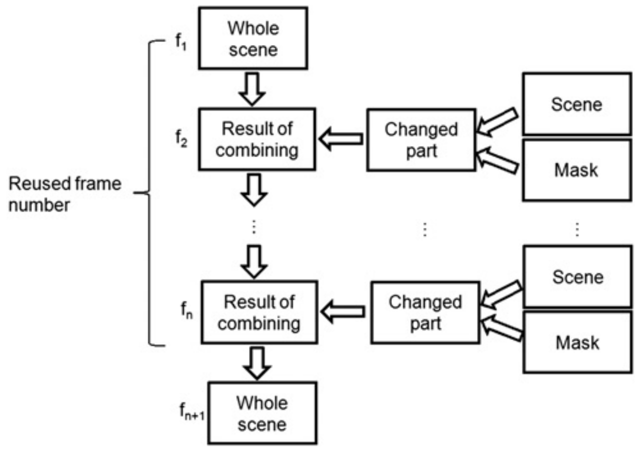

Fig. 4 *Reused frame number = n − 1*

4 Results

The proposed algorithm is run on a PC with 3.33 GHz Core™ i7 CPU and 8 GB of memory. The used graphics card is an NVDIA GeForce GTX 295. The proposed algorithm is implemented based on OpenGL and GLSL. All results are rendered at 1024 × 1024 pixels. The testing scenes are dragon, bunny, buddha and yeahRight with between 50 k and 755 k polygons. All scenes are defined in a box with one light source.

Most jobs are stored as textures for different rendering passes and are combined to obtain the rendering effects. The proposed adaptive sampling technique for volumetric shadows in dynamic scenes dedicates to reducing samples by reusing ray marching results from the precious frame. Therefore, at least two rendering passes are added to the rendering system. One is for building a stencil mask and another is for combining the reusable parts and the actually recomputed parts.

Table 1 shows the frame rates of the proposed temporally adaptive sampling approach and that of the hybrid ray marching [2]. In most scenes, such as dragon, bunny and buddha, the proposed approach can reach a definite speedup with the similar visual quality. However, the speedup is coming from reusing ray marching results from the previous frame. Therefore, the performance is directly proportional to moving objects and the shadow volume of the moving objects covering pixels.

Figs. 5–7 gain more speedups because of fewer pixels are changed, since we reuse most parts of the previous frame. Fig. 8 shows less speedup for scene yeahRight because of sophisticated model and the overhead for building shadow volumes.

Table 1 Results of scenes

Scene	Our approach, fps	Hybrid ray marching, fps	Speedup
dragon	46.08	35.24	1.31
bunny	59.93	44.73	1.34
buddha	58.76	41.63	1.41
yeahRight	29.34	26.75	1.10

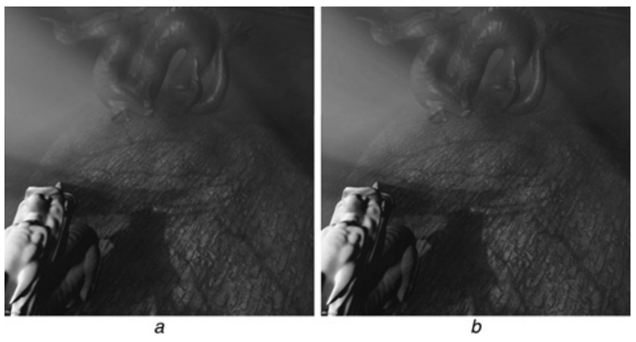

Fig. 5 *Scene 'dragon'*

a Generated by the proposed algorithm
b Generated by hybrid ray marching

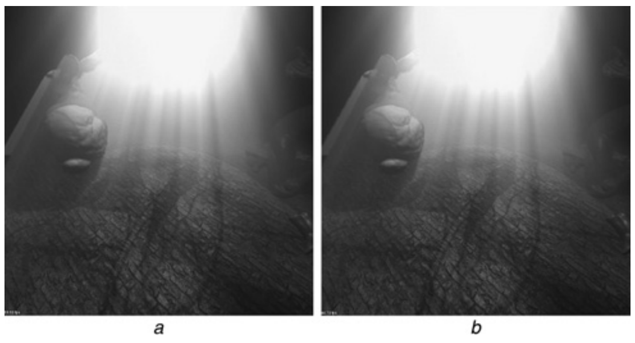

Fig. 6 *Scene 'bunny'*

a Generated by the proposed algorithm
b Generated by hybrid ray marching

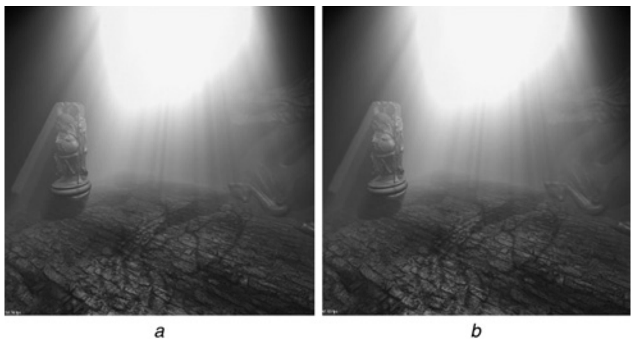

Fig. 7 *Scene 'buddha'*

a Generated by the proposed algorithm
b Generated by hybrid ray marching

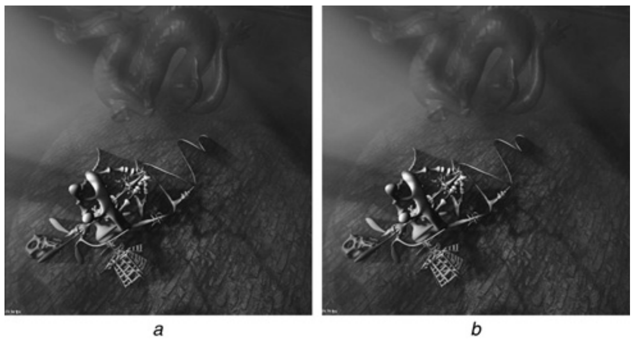

Fig. 8 *Scene 'yeahRight'*

a Generated by the proposed algorithm
b Generated by hybrid ray marching

There are three factors for affecting speedups: (a) the number of moving objects, (b) the stencil mask with occupied pixels and (c) the complexity of moving objects. All these factors exist when the performance of the proposed algorithm becomes slow. As shown in Table 1, the performance of scene yeahRight is influenced by the complexity of moving objects. Moreover, we capture the frames produced by the proposed algorithm and the hybrid ray marching. The difference images of these two frames show zeroes for all pixels if the frame rate is high.

Assume that we generate a background texture taking $T_{\text{background}}$ where the whole pixels are computed and a texture taking T_{reused} for only pixels covered by the stencil mask. Moreover, we update the whole scene by reusing a definite number of frames N_{reused}. Therefore, the average time complexity $T(n)$ of the proposed method for each frame is about

$$T(n) = \frac{T_{\text{background}} + T_{\text{reused}}(n-1)/N_{\text{reused}}}{n}$$

where n is the number of frames.

There are some limitations for the proposed approach. If the moving objects and their shadow volumes cover too many pixels of the screen, then speedup will be limited. Also, if the camera is keeping moving, there will be no pixels reusable. Therefore, there will be a little speedup or no speedup in these cases.

5 Conclusions and future works

In this paper, we propose an adaptive sampling approach for volumetric shadows in dynamic scenes. We achieve a definite speedup and maintain the similar visual quality. Besides, this method can be easily integrated to other rendering techniques since two rendering passes are only needed. In the future, we will generate the stencil mask more efficiently. Save the stencil mask of the objects of the previous frame and adopt it in the current frame when they are detected moving. Besides, a dynamic detection of changed parts of the stencil mask can be applied to decide whether using this sampling method or not. If moving objects and their shadow volumes cover over a definite percentage of the screen, we do not use this method.

6 References

1 Pharr, M., Humphreys, G.: 'Physically based rendering: from theory to implementation' (Morgan Kaufmann, 2004)
2 Wyman, C., Ramsey, S.: 'Interactive volumetric shadows in participating media with single-scattering'. Proc. IEEE Symp. on Interactive Ray Tracing, 2008, pp. 87–92
3 Engelhardt, T., Dachsbacher, C.: 'Epipolar sampling for shadows and crepuscular rays in participating media with single scattering'. Proc. 2010 ACM SIGGRAPH Symp. on Interactive 3D Graphics and Games, 2010, pp. 119–125
4 Pegoraro, V., Parker, S.: 'An analytical solution to single scattering in homogeneous participating media', *Comput. Graph. Forum*, 2009, **28**, (2), pp. 329–335
5 Sun, B., Ramamoorthi, R., Narasimhan, S., Nayar, S.: 'A practical analytic single scattering model for real time rendering', *ACM Trans. Graph.*, 2005, **24**, (3), pp. 1040–1049
6 Jarosz, W., Zwicker, M., Jensen, H.W.: 'The beam radiance estimate for volumetric photon mapping', *Comput. Graph. Forum*, 2008, **27**, (2), pp. 557–566
7 Jensen, H.W., Christensen, P.H.: 'Efficient simulation of light transport in scenes with participating media using photon maps'. Proc. SIGGRAPH 98, 1998, pp. 311–320
8 Sun, X., Zhou, K., Lin, S., Guo, B.: 'Line space gathering for single scattering in large scenes', *ACM Trans. Graph.*, 2010, **29**, (4), article no. 54
9 Max, N.L.: 'Atmospheric illumination and shadows'. Computer Graphics (Proc. SIGGRAPH '86), 1986, pp. 117–124
10 Dobashi, Y., Yamamoto, T., Nishita, T.: 'Interactive rendering method for displaying shafts of light'. Proc. Eighth Pacific Conf. on Computer Graphics and Applications, 2000
11 Imagire, T., Johan, H., Tamura, N., Nishita, T.: 'Anti-aliased and real-time rendering of scenes with light scattering effects', *Visual Comput.*, 2007, **23**, (9–11), pp. 935–944
12 Dobashi, Y., Yamamoto, T., Nishita, T.: 'Interactive rendering of atmospheric scattering effects using graphics hardware'. Proc. Graphics Hardware, 2002, pp. 99–107
13 Lefebvre, S., Guy, S.: 'Volumetric lighting and shadowing'. Available at http://www.aracknea-core.com/sylefeb/page.php?c=Shaders, 2002
14 Biri, V., Arques, D., Michelin, S.: 'Real time rendering of atmospheric scattering and volumetric shadows', *J. WSCG*, 2006, **14**, pp. 65–72
15 Mech, R.: 'Hardware-accelerated real-time rendering of gaseous phenomena', *J. Graph. Tools*, 2001, **6**, (3), pp. 1–16

16 Hu, W., Dong, Z., Ihrke, I., Grosch, T., Yuan, G., Seidel, H.P.: 'Interactive volume caustics in single-scattering media'. Proc. 2010 Symp. on Interactive 3D Graphics and Games, pp. 109–117

17 Billeter, M., Sintorn, E., Assarsson, U.: 'Real time volumetric shadows using polygonal light volumes'. Proc. High Performance Graphics, 2010

18 Baran, I., Chen, J., Ragan-Kelley, J., Durand, F., Lehtinen, J.: 'A hierarchical volumetric shadow algorithm for single scattering', *ACM Trans. Graph.*, 2010, **29**, (5), article no. 178

19 Chen, J., Baran, I., Durand, F., Jarosz, W.: 'Real-time volumetric shadows using 1D min–max mipmaps'. In ACM Symp. on Interactive 3D Graphics and Games, 2011

20 Wyman, C.: 'Voxelized shadow volumes'. ACM/EG Symp. on High Performance Graphics, 2011

21 Nishita, T., Miyawaki, Y., Nakamae, E.: 'A shading model for atmospheric scattering considering luminous distribution of light sources'. Proc. SIGGRAPH, 1987, pp. 303–310