Adaptive Depth Bias for Shadow Maps

Hang Dou [*1], Yajie Yan [†1], Ethan Kerzner [‡2], Zeng Dai [§1], and Chris Wyman [¶3]

[1]The University of Iowa [2]SCI Institute [3]NVIDIA

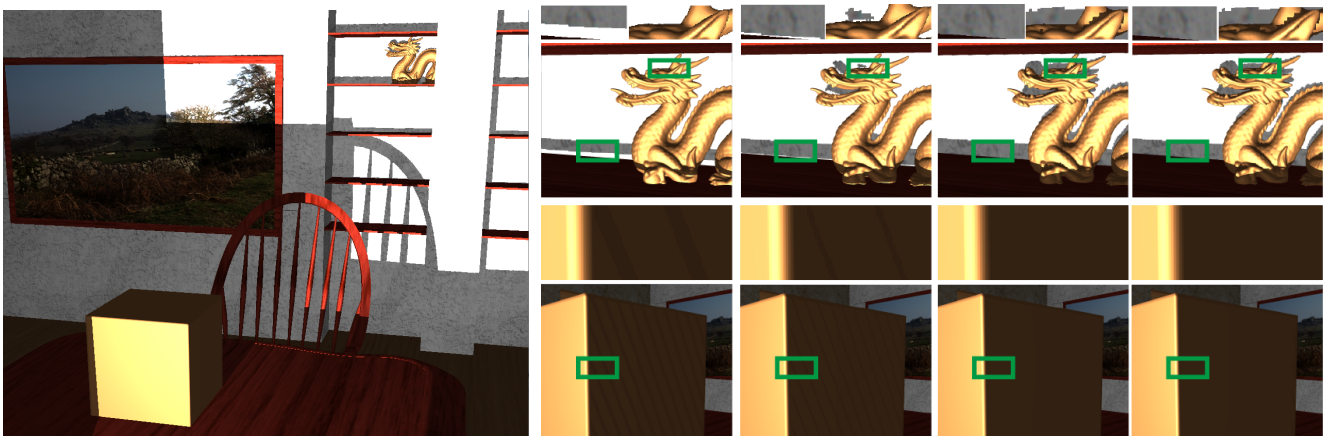

Figure 1: *The overview (left) of an interior scene illuminated by traditional shadow mapping and close look (right) to certain areas. Both constant depth bias (first column) and slope scale depth bias (second column) suffer from shadow acne and shadow detachment to different extent. Our method (third column) has no visible acne and preserves more shadow details. Dual depth layers depth bias (fourth column) is used as reference to compare our method against.*

Abstract

Shadow aliasing due to limited storage precision has been plaguing discrete shadowing algorithms for decades. We present a simple method to eliminate false self-shadowing through adaptive depth bias. Unlike existing methods which simply set the weight of the bias based on surface slope or utilize the second nearest surface, we evaluate the bound of bias for each fragment and compute the optimal bias within the bound. Our method introduces small overhead, preserves more shadow details than widely used constant bias and slope scale bias and works for common 2D shadow maps as well as 3D binary shadow volumes.

CR Categories: I.3.3 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing and texture

Keywords: bias, shadow, adaptive, real-time

[*]e-mail:hangdou@gmail.com
[†]e-mail:danielyan86129@gmail.com
[‡]e-mail: kerzner@sci.utah.edu
[§]e-mail:zeng-dai@uiowa.edu
[¶]e-mail:chris.wyman@acm.org

1 Introduction

Shadow mapping is the most commonly used method to cast surface shadows in games and other interactive applications. However, traditional shadow map suffers from shadow acne due to under sampled geometry in light space [Scherzer et al. 2011]. A constant depth bias [Williams 1978] can remove incorrect self-shadowing but also causes false unshadowing or shadow detachment. Non-constant depth methods [King 2004; Gautron et al. 2013] were proposed to avoid shadow detachment by restricting the bias amount based on surface slope scale. However, these methods fail to apply the minimal bias for each fragment and thus still produce false unshadowing to some extent.

Methods utilizing the second nearest surface depth [Woo 1992; Weiskopf and Ertl 2003] produce much less artifacts or acne than existing single depth layer based methods but sacrifice the performance due to the extra render pass, which makes them impractical in real time applications. Methods storing extra triangle information [Dietrich 2001; Dai et al. 2008] also reduce artifacts but they introduce more storage and have lower performance than traditional shadow map.

We propose a novel approach to remove false self-shadowing by generating bias for each fragment adaptively. We estimate a tight bias bound for each fragment and compute the optimal bias within the bound. We also introduce an adaptive epsilon to make sure the false shadowed fragment is shifted above its occluder. Our adaptive depth bias is easy to implement, comes with little cost and works well for 2D shadow maps and 3D binary shadow volumes in fully dynamic scenes. We apply our algorithm to traditional shadow map [Williams 1978], paraboloid shadow map [Brabec et al. 2002] and voxelized shadow volumes (VSVs) [Wyman 2011]. The main contributions of this paper are:

- analyzing the tight bias bound for each fragment;

- providing minimal depth bias to eliminate false self-shadowing.

2 Related Work

A detailed description of surface visibility algorithms is beyond the scope of this paper. Woo et al. [1990] and Eisemann et al. [2011] give nice surveys to shadowing algorithms. To remove false self-shadowing coming with shadow maps, research has been done from different aspects: warping shadow map texture, avoiding depth bias and using non-constant depth bias.

Methods were presented to parameterize the shadow map so that texture resolution is concentrated where it is needed. Cascaded shadow map [Tadamura et al. 2001] splits the view frustum into multiple frustum with different resolution. Adaptive shadow map [Fernando et al. 2001] replaces an entire shadow map texture with a hierarchical representation. Perspective shadow map [Stamminger and Drettakis 2002] and [Wimmer et al. 2004] warp the light frustum so that it only stores visibility information inside the view frustum. Our method can be viewed as a drop-in replacement for bias mechanism in these shadow maps.

Alias free shadow maps [Aila and Laine 2004] and irregular z-buffer [Gregory S. Johnson and Burns March 2004] are among the first to use eye-visible fragments as samples when generating shadow maps. However, the non-uniform nature of these methods lack direct support from the GPUs. Evolution of GPGPU techniques makes it possible to implement such ideas on the GPUs using CUDA [Arvo 2007; Sintorn et al. 2008]. Nonetheless, it is still not known how to achieve such flexibility by fully utilizing the GPUs' rendering pipeline.

A variety of research has explored methods to avoid depth bias. Wang and Molnar [1994] compare the depth between virtual samples and samples on second nearest surfaces. Using variance shadow map (VSM) does not need depth bias but still needs clamping the variance to a proper value before calculating Chebyshev's inequality. The main problem of these methods is that they introduce extra render pass, storage and multiple texture look-ups.

Gautron et al. [2013] generate the depth bias in proportion to the angle between the shaded fragment's surface normal and the light direction. King [2004] introduces slope scale depth bias which is also computed based on the fragment's normal direction and weighted by the depth slope from neighboring texels. These two methods do not provide minimal bias for each fragment. Compared to our approach, they have more false positive and true negative errors.

Woo et al. [1992] propose midpoint shadow mapping which uses depth difference of the closest and second closest surface as the depth bias. However, false self-shadowing remains an issue at silhouette lines. Weiskopf and Ertl [2003] improve this dual depth method by adding a constant bound to the midpoint depth bias. Dual depth layers based methods give a good result but introduce extra cost due to the extra rendering pass.

False self-shadowing also plagues shadow volume methods, especially binary shadow volumes. To avoid another look-up into the shadow map, Wyman and Dai [2011; 2013] use VSVs with a constant bias to cast surface shadows besides volumetric shadows. VSVs only store binary values and therefore the techniques mentioned above cannot be used to avoid false shadowing. Our method is applicable because it only depends on local geometry information.

3 Adaptive Bias for Traditional Shadow Maps

The amount of bias needed to eliminate false self-shadowing differs among fragments. To remove shadow acne with minimal false unshadowing, we first analyze the bound of depth bias for each fragment and then compute the optimal bias within the bound range. Subsequently, we shift the fragment using the optimal bias with an adaptive epsilon before visibility checking. Here we address the problem for traditional shadow map. Extension to voxelized shadow volumes will be discussed later.

3.1 Optimal Depth Bias

For a traditional shadow map, as shown in Figure 2 (left), fragment F_1 is shadowed by fragment F_2, whose depth is stored in the corresponding texel. We call F_2 the occluder of F_1 or occluder of the texel. Suppose F_1 and F_2 lie on the same planar surface P.

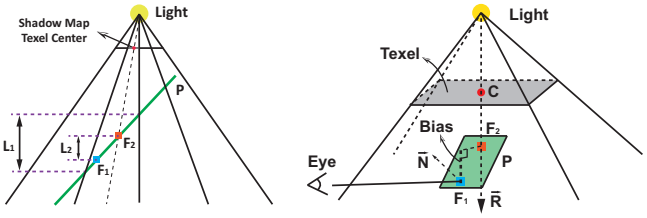

Figure 2: *(left) 2D illustration of bias bound for a given fragment lying on the planar surface. F_1 and F_2 are two fragments on planar surface P. L_1 represents the bound for a reasonable bias of F_1. L_2 represents the optimal bias for F_1. (right) 3D Illustration of adaptive bias computation for a traditional shadow map.*

Based on the observation that the shadow cast on F_1 only comes from the area covered by the corresponding texel, the bias for F_1 should be bounded by L_1. Any bias beyond L_1 may cause unexpected shadow detachment. The optimal bias is defined as L_2, which is the minimal bias needed to move F_1 above its occluder F_2. To compute this bias for a fragment, we first locate its potential occluder. As depicted in Figure 2 (right), given a fragment F_1, its potential occluder F_2 can be computed as the intersection of $\vec{R}$ and P, where $\vec{R}$ is the ray traced from the light source through the texel center C, and P is the tangent plane defined by F_1 and normal N. The optimal bias is then the depth difference between F_1 and F_2.

3.2 Adaptive Epsilon

To shift the fragment just above its potential occluder, a proper epsilon value is needed besides the optimal bias. However, a constant epsilon does not work since the depth value is stored non-linearly. Therefore, instead of using a constant epsilon directly, we transform the constant epsilon adaptively based on the depth compression function:

$$\epsilon = f'(x)\Delta x, \tag{1}$$

$$\Delta x = sceneScale \times K, \tag{2}$$

where ϵ denotes the adaptive epsilon, x represents the unnormalized depth value of fragment to be shaded, $f(x)$ is a depth compression function which maps depth values from near and far clipping plane distance to $[0, 1]$, Δx is the constant epsilon computed from $sceneScale$, the length of scene's bounding box diagonal and K, an empiric constant. In our implementation, we use standard

OpenGL depth compression function and obtain adaptive epsilon as follow:

$$\epsilon = \frac{(lf - depth \times (lf - ln))^2}{lf \times ln \times (lf - ln)} \times sceneScale \times K, \quad (3)$$

where ln and lf represent the light near and far plane distance, $depth$ represents the normalized depth value for the given fragment. We set $K = 0.0001$ in all our experiments.

3.3 Apply Adaptive Depth Bias

Once we have computed the optimal bias and the adaptive epsilon, we shift the fragment just above the potential occluder before visibility checking. Below we show the main steps of our algorithm.

```
SM ← generateShadowMap(LightPosition)
for each fragment F with normal N do
    P ← defineTangentPlane(F, N)
    C ← locateTexelCenter(SM, F)
    R ← defineLightRay(LightPosition, C)
    F′ ← planeRayIntersect(R, P)
    ϵ ← CalcAdaptiveEpsilon(F′)
    isLit ← checkVisibility(SM, F′, ϵ)
    outputColor ← isLit × shadeFrag(F)
end for
```

As illustrated in Figure 4 (right), our adaptive bias suffers from over bias when the shaded fragment's local tangent plane is almost parallel to the light ray, which results in unexpected noise. However, this only happens to the fragments whose local tangent plane almost parallel to light rays, which transports almost no radiance to the viewer with common materials like Lambertian and Phong.

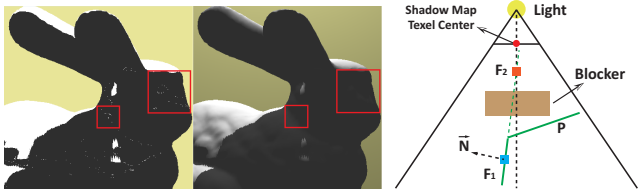

Figure 4: *(left) Our adaptive bias introduces noises because of overly shifting. (center) Noises are invisible under Lambertian shading. (right) Illustration of over shifting.*

Assuming fragments lying on planar surface provides a good approximation to the common situation in many real scenes. For hemispherical and omni directional light sources, paraboloid shadow mapping is the common choice. It is straightforward to apply our method to paraboloid shadow map. As shown in Figure 3, our method eliminated the false positive error while keeping most shadow details. Compared to slope scale depth bias [King 2004], we bound the bias in a smaller region and thus preserve more shadow details after shifting.

4 Adaptive Bias for Voxelized Shadow Volumes

Voxelized Shadow Volumes [Wyman 2011] compute both volume shadows in participating media and surface shadows. Similar to traditional shadowing algorithms, computing surface shadows with VSVs suffers from artifacts caused by discretization of geometry. Unlike traditional shadow maps, VSVs represent shadows with binary voxel grid. A voxel is occluded only if it contains an occluding fragment or it lies in the shadow of another occluding object. As shown in Figure 5, VSVs are defined in an epipolar space. To compute the adaptive bias for a fragment F_1, we obtain its voxel center $C_{V_1}(\alpha_c, \phi_c, \theta_c)$ as described in [Wyman 2011]. With C_{V_1}, we generate the shadow sample ray $\vec{R}$ to intersect with F_1's local tangent plane P for F_1's potential occluder F_2. Ideally we could move F_1 above F_2's voxel V_2 to remove false shadowing. However, this will cause false positive errors as shown in Figure 6. Assume V_2's adjacent voxel which is closer to light source is V_3. We use V_2 for visibility checking if F_2 is behind or on the voxel center. We use V_2 for visibility checking if F_2 is behind or on the voxel center. Below shows the shader pseudocode for our algorithm. Figure 7 shows the result of applying adaptive bias to VSVs.

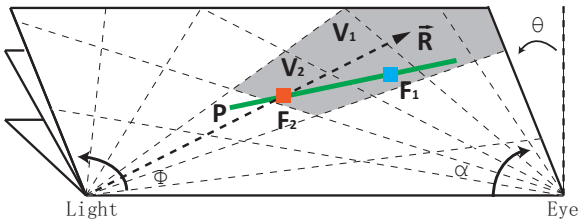

Figure 5: *Incorrect shadowing in the epipolar coordinate system for VSVs. Gray voxels mean occluded. The voxel containing fragment F_1 is incorrectly shadowed by the voxel containing fragment F_2.*

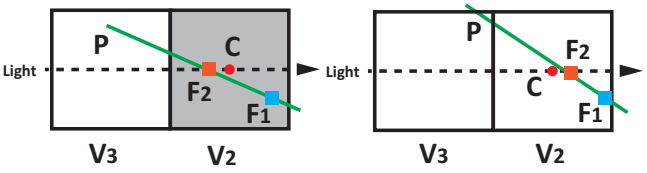

Figure 6: *V_3 and V_2 are two adjacent voxels. C is V_2's voxel center. F_1 and F_2 lie on the planar surface P. (left) V_2 is occluded. Using V_2 for visibility checking will cause false self-shadowing of F_1. (right) V_2 is lit. Use V_2 for visibility checking won't cause false self-shadowing.*

```
VSVs ← GenerateVSVs(LightPosition, eyePosition)
for each fragment F with normal N do
    P ← defineTangentPlane(F, N)
    C_V ← locateVoxelCenter(VSVs, F)
    R ← defineLightRay(LightPosition, C_V)
    F′ ← planeRayIntersect(R, P)
    C′_V ← locateVoxelCenter(VSVs, F′)
    C″_V ← adjacentCloserToLightVoxel(C′_V)
    if F′ is behind or on C′_V then
        isLit ← checkVisibility(VSVs, C′_V)
    else
        isLit ← checkVisibility(VSVs, C″_V)
    end if
    outputColor ← isLit × shadeFrag(F)
end for
```

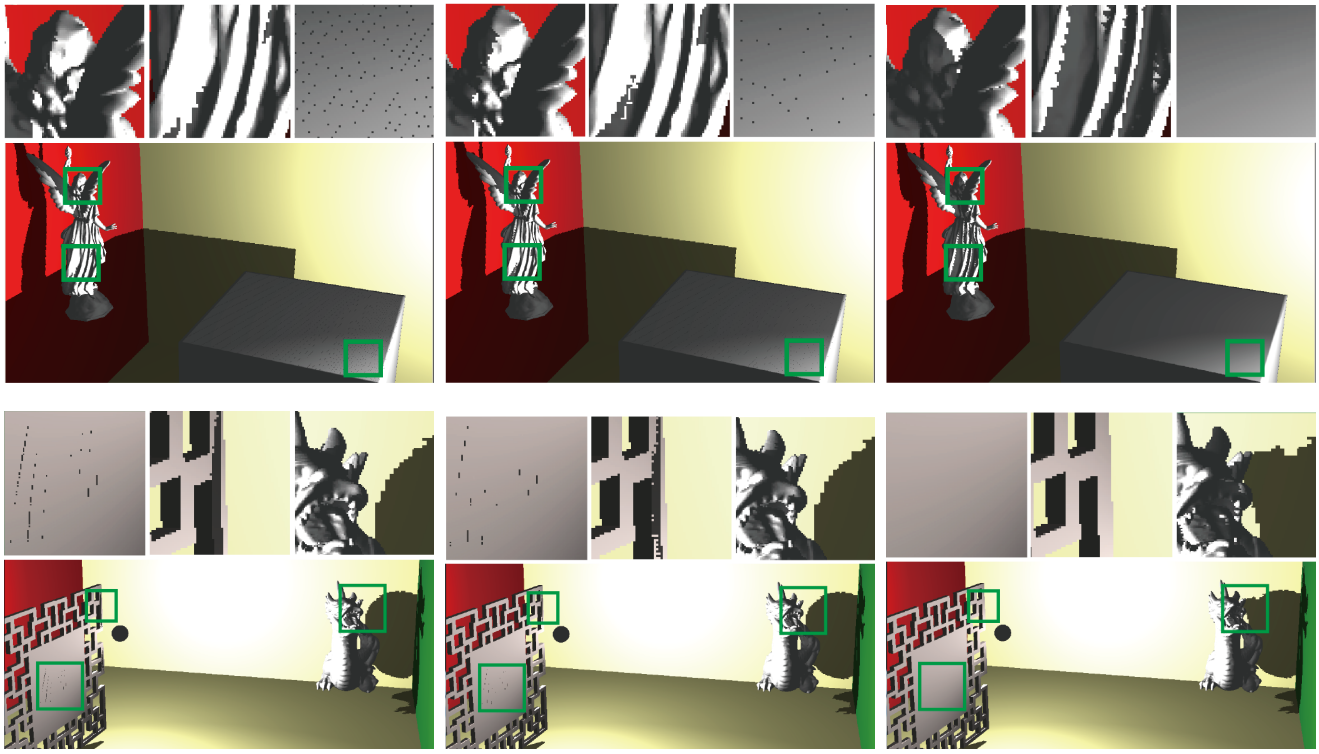

Figure 3: *Comparison between our depth bias algorithm and commonly used depth bias algorithms in traditional shadow map (top) and paraboloid shadow map (bottom). Constant depth bias (left) and slope scale depth bias (center) both suffer from false shadowing. Our method (right) has no visible shadow acne and preserves most shadow details.*

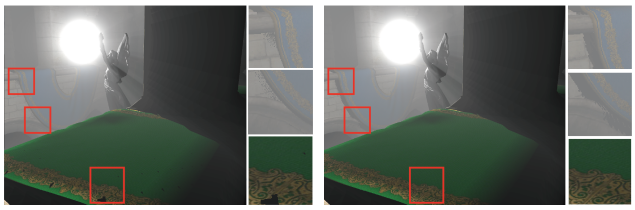

Figure 7: *Comparison between our adaptive depth bias (right) and constant depth bias (left) applied respectively to VSVs. Our adaptive bias saves more shadow details while erasing shadow acne.*

Method	Shadow Map	Final Shading	Overall
Constant	2.095ms	4.232ms	6.327ms
Slope Scale	2.112ms	4.535ms	6.647ms
Ours	2.108ms	5.211ms	7.319ms
Dual Layer	4.716ms	5.174ms	9.890ms

Table 1: *Performance measure of Sponza scene (20K polygons). The scene is lit through traditional shadow mapping with a shadow map resolution of* 1024×1024.

5 Results and Discussion

We implement our method through OpenGL/GLSL in C++ and the test scenes are rendered on a machine with Intel(R) Cores(TM) i7 CPU @2.93GHz and a NVIDIA graphics card GTX580. The output resolution of all the generated images is 1024×1024. In our test scenes, VSVs were generated by applying shadow map resampling and prefix scan as in [Wyman 2011].

Figure 8 and Figure 9 compare our method with constant depth bias and slope scale depth bias [King 2004] for traditional shadow map and paraboloid shadow map in a complex scene. We use dual depth layers method [Weiskopf and Ertl 2003] as reference images. With a constant bias, objects close to the light still have shadow acne left while objects far from the light already suffer from heavy shadow detachment. Slope scale depth method weights the bias by surface slope scale and gets less false unshadowing. However, the constant part still results in visible shadow detachment in a large dynamic scene. In our test scenes, our method gives results equivalent to dual depth shadow mapping but with significant improved performance.

Table 1 shows the corresponding performance of the scene depicted in Figure 8. While constant depth bias adds no overhead, slope scale depth bias adds very little overhead to overall rendering time. Dual layers depth bias needs two rendering passes and thus double the time necessary for shadow map generation. Besides, an extra texture look-up costs close to 0.6ms, consuming 18% more rendering time compared with constant depth bias. In the shading stage, cost from computing adaptive bias in our method is close to the cost from an extra texture look-up in dual layers based method. However, with an extra render pass in shadow map generation stage, dual layer based method costs close to 50% more rendering time compared with our method.

Figure 11 shows the performance chart of the scene in Figure 9 (19M polygons). As the shadow map resolution increases, slope scale depth costs around 5% more rendering time compared with constant depth bias. Our method costs around 20% more rendering

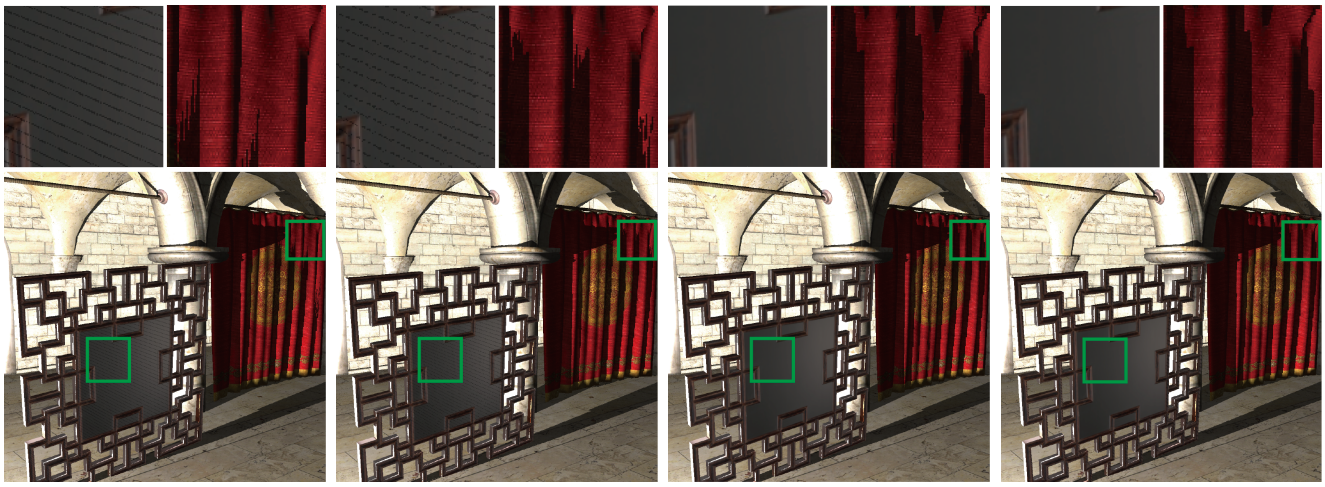

Figure 8: *Sponza illuminated by a traditional light source with a shadow map resolution of* 1024×1024*. (left) Constant depth bias and slope scale depth bias (center left) suffer from shadow acne and shadow detachment to some extent. Our method (center right) has no visible false shadows. Dual depth layers depth bias (right) serves as a reference image.*

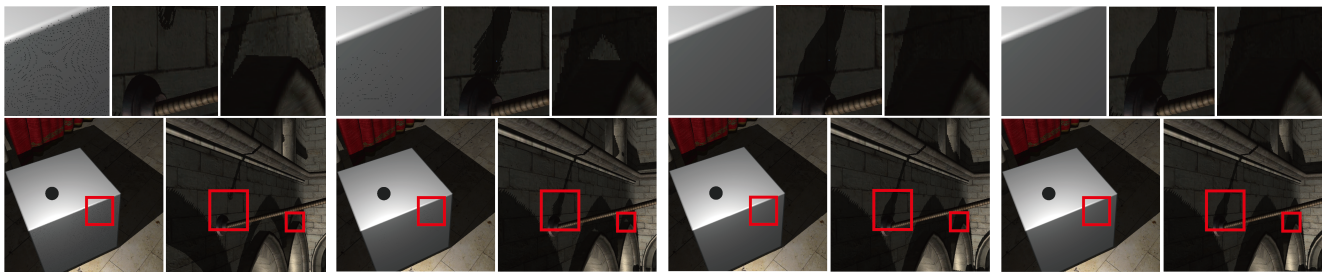

Figure 9: *The scene is lit by an omni-directional light source through dual paraboloid shadow mapping with a shadow map resolution of* 2048×2048*. We move the camera to different parts of the scene. (left) Constant depth bias and slope scale depth bias (center left) both leave shadow acne on the cube and suffer from false unshadowing on the wall to some extent. Our method (center right) eliminates all the shadow acne on the cube without shadow detachment on the wall. Dual depth layers depth bias (right) serves as a reference image.*

Figure 10: *The scene is lit by a omni-directional light source through VSVs with a volume resolution of* $1024 \times 1024 \times 512$*. (left) Surface shadows are cast through VSVs with a constant bias. (center) Surface shadows are cast through VSVs with our adaptive bias. (right) Reference image. Surface shadows are cast with a 2D* 8192×8192 *shadow map.*

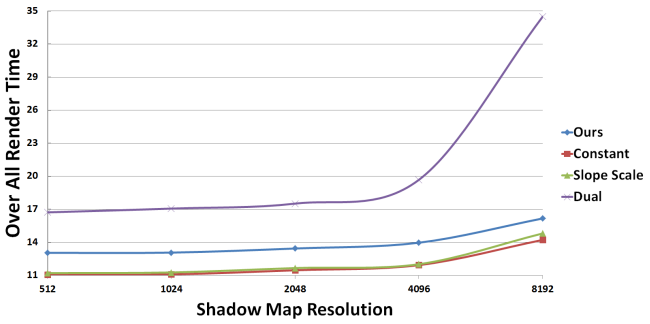

Figure 11: *Performance comparison of different depth bias algorithms under different shadow map resolution. The rendering time is measured in milliseconds.*

time compared with constant bias with much less false shadowing. Compare to our method, dual layer depth map gives a nearly equivalent image quality but it costs 50% more rendering time when shadow map resolution is under 4096×4096 and costs more than 50% when shadow map resolution keeps increasing.

Figure 10 shows the result of applying our adaptive bias to VSVs. Since VSVs only contain binary value in each voxel, existing depth bias algorithms, such as slope scale depth bias and dual depth layers can not be used to eliminate false self-shadowing. So we only compare the rendering result of adaptive bias with constant bias. VSVs' nature of non-uniform epipolar voxel grid and view-dependence make constant bias hard to work well. The constant bias is set so that there remains some false self-shadowing on the arch while already suffering some shadow detachment near the distant blue curtain. Our adaptive depth bias reduces the false self-shadowing while preserving more shadow details than fixed constant depth bias.

6 Conclusion

We proposed a method to eliminate false self-shadowing for surface shadowing algorithms by producing adaptive depth bias. We confine the depth bias for each fragment within a tight bound and compute the minimal bias needed to eliminate shadow acne. We implemented our algorithm for 2D shadow maps as well as 3D voxelized shadow volumes. Compared to constant depth bias and slope scale depth bias, our adaptive bias removes more false self-shadowing and causes less false unshadowing. Our method gives an equivalent result as dual depth layers based method but with much higher performance.

References

AILA, T., AND LAINE, S. 2004. Alias-free shadow maps. In *Proceedings of Eurographics Symposium on Rendering*, vol. 2004, 161–166.

ARVO, J. 2007. Alias-free shadow maps using graphics hardware. *Journal of Graphics, GPU, and Game Tools 12*, 1, 47–59.

BRABEC, S., ANNEN, T., AND SEIDEL, H.-P. 2002. Shadow mapping for hemispherical and omnidirectional light sources. In *Advances in Modelling, Animation and Rendering*. Springer, 397–407.

DAI, Q., YANG, B., AND FENG, J. 2008. Reconstructable geometry shadow maps. In *Proceedings of the 2008 symposium on Interactive 3D graphics and games*, ACM, 4.

DIETRICH, S. 2001. Practical priority buffer shadows. *Game Programming Gems II*, 482.

EISEMANN, E., SCHWARZ, M., ASSARSSON, U., AND WIMMER, M. 2011. *Real-time shadows*. AK Peters, Ltd.

FERNANDO, R., FERNANDEZ, S., BALA, K., AND GREENBERG, D. P. 2001. Adaptive shadow maps. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, ACM, 387–390.

GAUTRON, P., MARVIE, J., AND BRIAND, G., 2013. Method for generating shadows in an image, Jan. 23. EP Patent 2,411,967.

GREGORY S. JOHNSON, W. R. M., AND BURNS, C. A. March 2004. The irregular z-buffer and its application to shadow mapping. Tech. rep., University of Texas Department of Computer Sciences Technical Report.

KING, G. 2004. Shadow mapping algorithms. *GPU Jackpot presentation*, 354–355.

SCHERZER, D., WIMMER, M., AND PURGATHOFER, W. 2011. A survey of real-time hard shadow mapping methods. In *Computer Graphics Forum*, vol. 30, Wiley Online Library, 169–186.

SINTORN, E., EISEMANN, E., AND ASSARSSON, U. 2008. Sample based visibility for soft shadows using alias-free shadow maps. *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering 2008) 27*, 4, 1285–1292.

STAMMINGER, M., AND DRETTAKIS, G. 2002. Perspective shadow maps. In *ACM Transactions on Graphics (TOG)*, vol. 21, ACM, 557–562.

TADAMURA, K., QIN, X., JIAO, G., AND NAKAMAE, E. 2001. Rendering optimal solar shadows with plural sunlight depth buffers. *The Visual Computer 17*, 2, 76–90.

WANG, Y., AND MOLNAR, S. 1994. Second-depth shadow mapping. *UNC-CS Technical Report TR94-019*.

WEISKOPF, D., AND ERTL, T. 2003. Shadow mapping based on dual depth layers. In *Proceedings of Eurographics*, vol. 3, 53–60.

WILLIAMS, L. 1978. Casting curved shadows on curved surfaces. In *ACM SIGGRAPH Computer Graphics*, vol. 12, ACM, 270–274.

WIMMER, M., SCHERZER, D., AND PURGATHOFER, W. 2004. Light space perspective shadow maps. In *Proceedings of the Fifteenth Eurographics conference on Rendering Techniques*, Eurographics Association, 143–151.

WOO, A., POULIN, P., AND FOURNIER, A. 1990. A survey of shadow algorithms. *Computer Graphics and Applications, IEEE 10*, 6, 13–32.

WOO, A. 1992. The shadow depth map revisited. In *Graphics Gems III*, Academic Press Professional, Inc., 338–342.

WYMAN, C., AND DAI, Z. 2013. Imperfect voxelized shadow volumes. In *ACM SIGGRAPH 2013 Talks*, ACM, New York, NY, USA, SIGGRAPH '13, 18:1–18:1.

WYMAN, C. 2011. Voxelized shadow volumes. In *Proceedings of the ACM SIGGRAPH Symposium on High Performance Graphics*, ACM, 33–40.