Real-time Rendering of Bumpmap Shadows Taking Account of Surface Curvature

Koichi Onoue
The University of Tokyo
5-1-5 Kashiwanoha,Kashiwa,
Chiba, Japan
Phone: +81.4.7136.3946
Fax: +81.4.7136.3943
onoue@is.s.u-tokyo.ac.jp

Nelson Max
University of California, Davis
P. O. Box 808, L-560,
Livermore, CA 94551, U.S.A
Phone: +1.925.422.4074
Fax: +1.925.422.6287
max2@llnl.gov

Tomoyuki Nishita
The University of Tokyo
5-1-5 Kashiwanoha,Kashiwa,
Chiba, Japan
Phone: +81.4.7136.3942
Fax: +81.4.7136.3943
nis@is.s.u-tokyo.ac.jp

Abstract

The bump-mapping technique is often used to represent bumps on objects such as bark on trees and craters on the moon. In order to render shadows cast by bumps, the horizon map method was proposed. The horizon map is a table which has, for each of a small collection of azimuthal directions, slopes from each viewpoint on the bump map (height field) to the corresponding horizon point, which is the highest viewable point seen from that viewpoint. In this paper, we propose a more precise method for rendering bumpmap shadows using a both a horizon map and a distance map, to take curvature of surfaces into consideration. The distance map is a table which has, for each azimuthal direction, horizontal projected distances from each point of the bump map to its corresponding horizon point. The proposed method can render shadows efficiently by using programmable graphics hardware.

keywords: *shadows, bump-mapping, horizon map, curvature, graphics hardware*

1 Introduction

In order to render realistic images, shadows are indispensable. In the field of computer graphics, many researchers still study shadow rendering methods [1]. On the other hand, bump mapping [2], which is a method to represent and render bumps efficiently, is widely used. Furthermore, since the performance of graphics hardware (GPU) has increased rapidly, bump mapping can now be processed on the graphics hardware. There has also been previous work on the shadows of bumps. Max [3] proposed the horizon map to render shadows cast by bumps. Noma proposed a rendering method of shadows on bump-mapped surfaces cast by other objects [4]. Forsyth [5] used a three-dimensional texture to implement Max's method in graphics hardware. However his method did not take account of surface curvature. Shadows of the bumps on bump-mapped surfaces are called bumpmap shadows in the rest of this paper.

In this paper, we propose to render bumpmap shadows more precisely by taking account of surface curvature. When bump mapping is used, surfaces facing away from the light source are not lit, and the shading process is usually skipped. However if the bumps on a curved surface are tall enough, parts of the bumps on the back-facing surfaces may be lit. Similar cases of the illumination of bumps beyond the shadow terminator were considered by Koenderink et al. [6]. The presentation of that paper inspired our implementation, which can render such effects in graphics hardware using a programmable shader.

The rest of this paper is organized as follows. First, related work is introduced in Section 2, and the extended algorithm for horizon mapping is explained in Section 3. Rendering results of bumpmap shadows are shown in Section 4. Conclusions and future work are described in Section 5.

2 Related Work

Max [3] proposed horizon mapping as a method of adding a shadow effect to the bump mapping that Blinn [2] had proposed. The horizon map is a table, which has slopes from each viewpoint on the bump map (height field) to the highest point seen from that viewpoint. Shadows are calculated by comparing the slopes stored in the horizon map with the slope of the ray to the light source.

Sloan et al. [7] proposed a method to implement horizon mapping in graphics hardware. They store horizon map as multiple two-dimensional textures indexed by the azmuthal

angle, and render shadows by interpolating between the two appropriate textures using multiple rendering passes. Kautz et al. [8, 9] approximated, with an elliptical cone, the set of light directions which can illuminate each point of a bump map. Shadows are rendered by detecting whether the light direction is included the elliptical cone. These methods need multiple rendering passes, so rendering costs are high.

The other method to represent bumps is displacement mapping [10]. This method subdivides the surfaces of objects and displaces the generated vertices. In order to render the shadows of the bumps, displacement mapping can be combined with rendering methods for geometry-based shadows, such as the shadow map method [11, 12] and the shadow volume method [13]. However, this subdivision greatly increases the number of vertices and polygons, and therefore their associated access, transformation, and set-up costs in the graphics hardware. Wang et al. [14] proposed a displacement mapping method without increasing the number of polygons. They store as textures visibility information for a height field seen from some sample directions, and do take some account of curvature. However their method needs a lot of memory to store the textures. Although they propose a compression method for the visibility information, the amount of memory consumption is still much greater than that of the horizon map.

Forsyth [5] proposed a method to implement the horizon map by using a three-dimensional texture, which is now supported by commodity graphics hardware. Bumpmap shadows can be rendered in one pass by his method, which is simple and easy to implement. However, Forsyth's method does not consider the surface curvature of objects so the shapes of the shadows are not accurate.

In this paper, we propose a method for rendering bumpmap shadows which is an extension of Forsyth's method and takes account of surface curvature.

3 Bumpmap Shadows

In this paper, as in Max [3], we use a local coordinate system whose axes are $P_u, P_v, N/|N|^{1/2}$. Here, P_u and P_v are partial derivative vectors along texture coordinates, and $N = P_u \times P_v$. Let L' the light vector L converted into this coordinate system, θ be an angle between L' and N, and φ be an angle from P_u when L' is projected onto the (P_u, P_v) plane.

Forsyth implemented the horizon mapping by storing the horizon map $\beta(u, v, \varphi)$ as three-dimensional texture. Here, we use in addition a distance map $d(u, v, \varphi)$ storing horizontal distances between the projections of $S(u, v)$ and its horizon point Q, as shown in Fig. 1. In short, three tables are used: the bump map $h(u, v)$, the distance map $d(u, v, \varphi)$, and the horizon map $\beta(u, v, \varphi)$. These tables are stored on the GPU memory as textures. The table $h(u, v)$ is used as in [2] to determine the bump shading, and all three tables are used for the bump shadows.

3.1 Local Curvature

The proposed method renders bumpmap shadows taking account of surface curvature. Our method of calculating the principal curvatures at a point is the same as the one used in Wang et al.'s paper [14].

For each vertex, principal curvatures κ_1, κ_2 and principal directions φ_1, φ_2 on the (u, v) plane are calculated. The vectors V_1 and V_2 on the surface in the principal directions are perpendicular on a curved surface, but their projections are not always perpendicular on the (u, v) plane. Therefore, in order to calculate the local curvature in a direction V which is the projection of L' onto tangent plane at P, the following method is used.

First, we precompute V_1 and V_2 at each vertex of the polygonal surface. Then V_1 and V_2 at P are calculated at each pixel fragment by interpolation. After calculating $\cos\varphi = (V \cdot V_1)/(|V||V_1|)$ and $\sin\varphi = (V \cdot V_2)/(|V||V_2|)$, we use the following formula:

$$\kappa = \kappa_1 \cos^2\varphi + \kappa_2 \sin^2\varphi. \quad (1)$$

Note that φ need not be calculated, only the expressions $\cos^2\varphi$ and $\sin^2\varphi$ need to be calculated. These expressions use squares of $\cos\varphi$ and $\sin\varphi$, so the square root need not be calculated when normalizing the vectors. That is, instead of $|V|, |V_1|, |V_2|$, only $|V|^2, |V_1|^2, |V_2|^2$ are calculated.

The radius of curvature R at P in direction V, that is, the radius of curvature of the intersection curve of the plane through P including N and L', and the curved surface $S(u, v)$, is given by $R = 1/\kappa$.

3.2 Horizon Angles Taking Account of Curvature

The calculation of the horizon angle β taking account of curvature is considered in the following two cases.

First, in the case shown in Fig. 2, the horizon angle is calculated by using the distance d to the horizon Q. Let $H = d\tan\beta$ be the difference of the height at Q and that at P (see Fig. 1). We calculate a decrease of the height at the horizon Q according to curvature of the surface as follows (see Fig. 2):

$$\begin{aligned} d &= \gamma R, \gamma = \frac{d}{R} = \kappa d \\ \overline{CA} &= 2(R+h)\sin\frac{\gamma}{2} \\ \overline{BC} &= \overline{CA}\cos\frac{\gamma}{2} = 2(R+h)\sin\frac{\gamma}{2}\cos\frac{\gamma}{2} \\ \overline{AB} &= \overline{CA}\sin\frac{\gamma}{2} = 2(R+h)\sin^2\frac{\gamma}{2} \end{aligned}$$

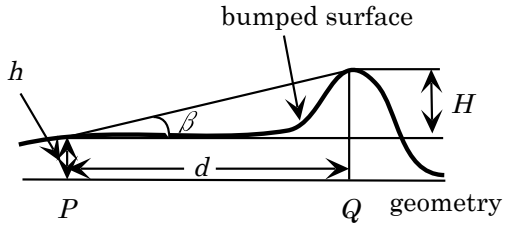


Figure 1. Relation of a point P on bump map and the horizon Q.

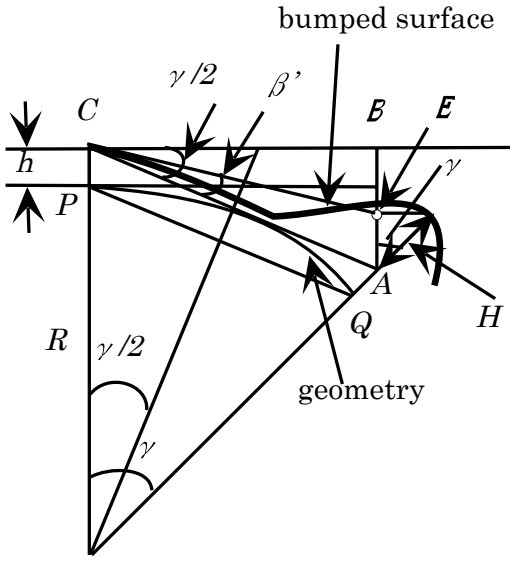


Figure 2. Change of height at the horizon Q by curvature.

$$\begin{aligned}\overline{EA} &= H\cos\gamma = d\tan\beta\cos\gamma \\ \overline{BE} &= \overline{EA} - \overline{AB} \\ &= d\tan\beta\cos\gamma - 2(R+h)\sin^2\frac{\gamma}{2}.\end{aligned}$$

Therefore, β' is modified like this:

$$\begin{aligned}\beta' &= arctan\frac{\overline{BE}}{\overline{BC}} \\ &= arctan\left(\frac{d\tan\beta\cos\gamma - 2(R+h)\sin^2\frac{\gamma}{2}}{2(R+h)\sin\frac{\gamma}{2}\cos\frac{\gamma}{2}}\right).\end{aligned}$$

When γ is small, $\cos\gamma \approx 1, \cos\frac{\gamma}{2} \approx 1, \sin\gamma \approx \gamma, \sin\frac{\gamma}{2} \approx \frac{\gamma}{2}$, therefore,

$$\beta' \approx arctan\left(\frac{d\tan\beta - 2(R+h)\frac{\gamma^2}{4}}{2(R+h)\frac{\gamma}{2}}\right)$$

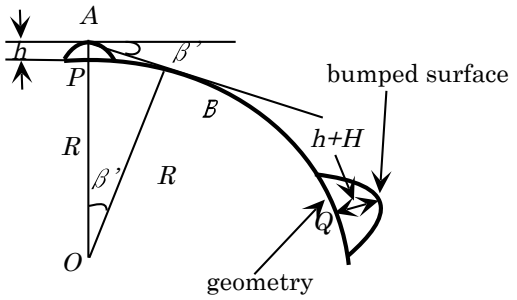


Figure 3. Horizon Q is under the horizon of curved surface.

$$= arctan\left(\frac{d\tan\beta - \frac{1}{2}(R+h)\gamma^2}{(R+h)\gamma}\right).$$

In the horizon map, we store $\delta = \tan\beta$ instead of β. The arc length is $d = R\gamma$, so

$$\begin{aligned}\delta' &= \tan\beta' = \frac{d\tan\beta - \frac{1}{2}(R+h)\gamma^2}{(R+h)\gamma} \\ &= \frac{d\tan\beta}{(R+h)\gamma} - \frac{1}{2}\gamma = \frac{R\gamma\tan\beta}{R\gamma(1+\frac{h}{R})} - \frac{1}{2}\gamma \\ &\approx \tan\beta(1-\frac{h}{R}) - \frac{1}{2}\gamma,\end{aligned}$$

or,

$$\tan\beta' \approx \tan\beta(1-\kappa h) - \frac{1}{2}\kappa d. \qquad (2)$$

The second case is the one considered by Koenderink et al. [6], when tall bumps are lit even when they are beyond the terminator, on a part of the surface facing away from the light source. As shown in Fig. 3, when the horizon point Q is under the horizon of the curved surface (the geometry of the object without the bumps), we calculate β' by the following method. In this case, d and β are not used to calculate β'. The proposed method does not store information about the height of the non-bumped horizon point B shown in Fig. 3, so we regard the height of B as 0.

The values of $\cos\beta'$ and $\sin\beta'$ in Fig. 3 are

$$\cos\beta' = \frac{R}{h+R} = \frac{1}{\frac{h}{R}+1} = \frac{1}{\kappa h+1}$$

$$\sin\beta' = -\sqrt{1-\cos^2\beta'},$$

therefore

$$\tan\beta' = \frac{\sin\beta'}{\cos\beta'} = -\frac{\sqrt{1-\cos^2\beta'}}{\cos\beta'}. \qquad (3)$$

In practice, we use the larger of the two values from equations (2) and (3) as the value of $\tan\beta'$, since a point on a bump could be shadowed either by a specific bump at distance d as in figure 2, or by the horizon of the curved non-bumped surface, as shown in figure 3.

All three tables β, h, d prepared in the proposed method are not needed, because $\tan\beta$ can be calculated by the expression:

$$\tan\beta = \delta = h((u,v) + d\,\hat{\varphi})/d, \tag{4}$$

where $\hat{\varphi}$ is a unit vector in the direction of φ. However, if only the textures h and d are prepared, more calculations are needed. Moreover, the height by interpolation from the table h does always coincide with the height of a point which is calculated by using d and β.

4 Results

Fig. 4 shows a part of a sphere bump-mapped with cylindrical columns. Bumpmap shadows are also rendered in this figure. Fig. 4(a) is the result rendered by the previous method [5], and Fig. 4(b) is the result of the proposed method. The light illuminates parts of the surface facing away from the light source (on the upper left side of the figures) due to equation (3), and the shadows of the bumps onto the smooth part of the surface in the proposed method are shorter than that in the previous method, due to equation (2). A part of a sphere with smaller curvature (larger radius) is rendered in Fig. 5. Comparing Fig. 4 with Fig. 5 shows that differences between the proposed method and the previous method become more clear as curvature become larger. The sphere consists of 8,192 polygons. The size of the bump map is 128×128, and the number of sample directions used to create the horizon/distance map is 32. In this example, 32 sample directions are needed to render shadows smoothly. The frame rate is about 25 fps(frames per second). The GPU memory used here is about 1.0MB.

A Bezier surface bump-mapped with reptilian skin is shown in Figs.6 and 7. The heights of the reptilian skin are inverted in Fig. 7. In this case, the shadow region of the proposed method is also smaller than that of the previous method. The Bezier surface is rendered as 512 polygons, and the frame rate is about 25 fps. The size of the bump map is 256×256, and the number of sample directions used to create the horizon/distance map is 16. The GPU memory used here is about 2.2MB.

In order to evaluate our proposed method, we compare bumpmap shadows with shadows of a displacement-mapped polygonal object rendered by the shadow volume method. Fig. 8(c) is the rendering result of accurate shadows by using displacement mapping and the shadow volume method, Fig. 8(b) is the result of the proposed method, and Fig. 8(a) is a result of previous method. Although the shadows in Fig. 8(a) are longer than Fig. 8(c), the shadows in Fig. 8(b) are as long as in Fig. 8(c). Therefore the proposed method can render shadows more accurately than the previous method. The displacement-mapped object consists of 77,760 polygons, while the bump-mapped object consists of 960 polygons. This difference results in different frame rates. The frame rates of Figs.8(a), (b), and (c) are 75, 75, and 15 fps, respectively. Therefore the proposed method can render accurate shadows faster than the shadow volume method applied to the displacement-mapped object. Moreover, the bumps are rendered more smoothly in Fig. 8(b) than in Fig. 8(c), that is, artifacts can be seen in Fig. 8(c). The displacement-mapped object needs more polygons to be rendered precisely.

The frame rates of the results shown in this section are measured by using Dell Precision 360 (CPU: Pentium4 3.0GHz, main memory: 1.0GB, GPU: Quadro FX 3000, GPU memory: 256MB). The proposed method is implemented by using OpenGL and Cg.

5 Conclusion and Future Work

In this paper, we have proposed a rendering method for bumpmap shadows taking account of curvature by using the horizon map and the distance map. The proposed algorithm can be implemented by a programmable shader in graphics hardware, and can be rendered at an interactive frame rate. The effects of the proposed method stand out when a bump map with a large height is mapped onto a surface with large curvature.

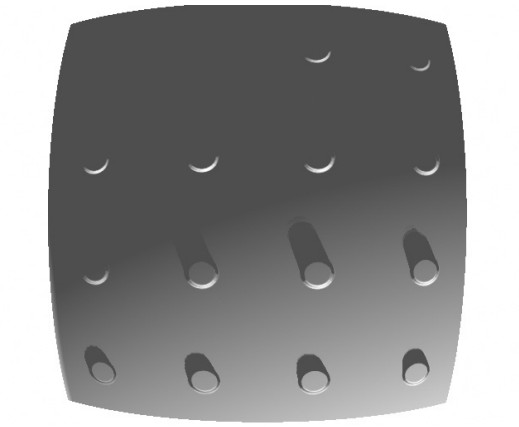
The proposed method focuses only on the local shadows around the bumps. Therefore it cannot render shadows cast by distant object profiles. Although such shadows can be rendered by the shadow map method or the shadow volume method, these methods cannot consider the bumps at the profile edges of bump-mapped objects as viewed from a light source. Representing such shadows is left as future work.

References

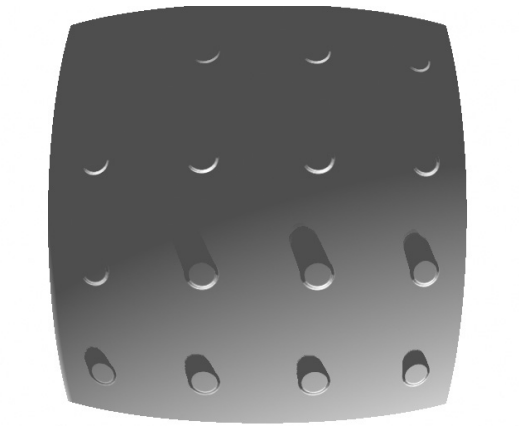
[1] A. Woo, P. Poulin, and A. Fournier. A survey of shadow algorithms. *IEEE Computer Graphics and Applications*, 10(6):13–32, Nov. 1990.

[2] J. F. Blinn. Simulation of wrinkled surfaces. In *Proceedings of the 5th annual conference on Computer graphics and interactive techniques*, pages 286–292. ACM Press, 1978.

[3] N. L. Max. Horizon mapping: shadows for bump-mapped surfaces. *The Visual Computer*, 4(2):109–117, 1988.

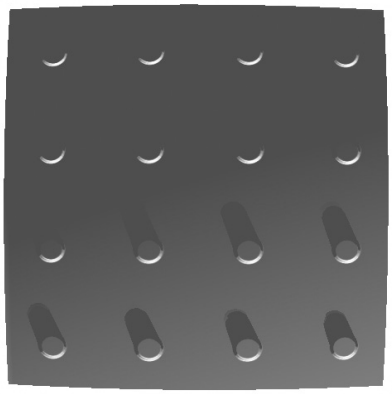


(a) bumpmap shadows by the previous method

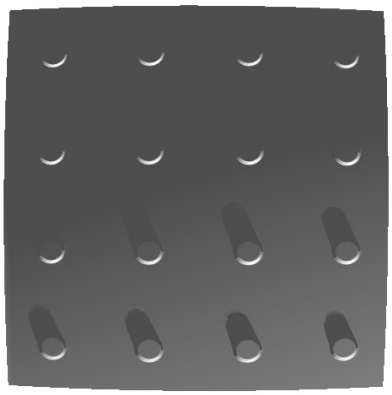


(b) bumpmap shadows by the proposed method

Figure 4. Bumpmap shadows on a sphere with 0.015 curvature.



(a) bumpmap shadows by the previous method



(b) bumpmap shadows by the proposed method

Figure 5. Bumpmap shadows on a sphere with 0.005 curvature.

[4] T. Noma and K. Sumi. Shadows on bump-mapped surfaces. *The Visual Computer*, 10(4):330–336, 1994.

[5] T. Forsyth. Self-shadowing bumpmap using 3d texture hardware. *Journal of Graphics Tools*, 7(4):19-26, 2003.

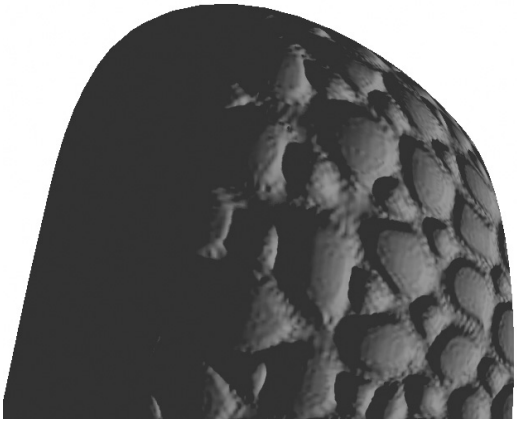
[6] J. J. Koenderink and S. C. Pont. Texture at the terminator. *1st International Symposium on 3D Data Processing Visualization and Transmission*, pages 406-415, Jun, 2002.

[7] P. P. Sloan and M. F. Cohen. Interactive horizon mapping. *Eurographics Workshop on Rendering*, pages 281–286, 2000.

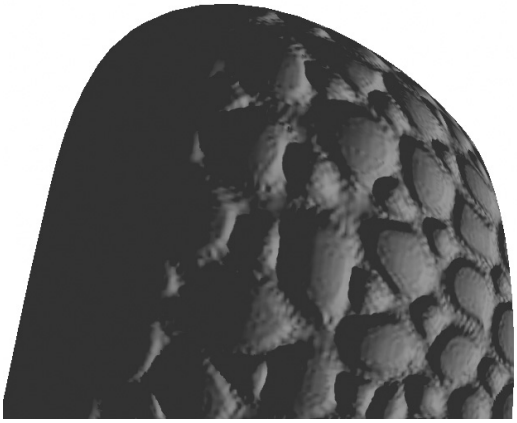
[8] J. Kautz, W. Heidrich, and K. Daubert. Bump map shadows for opengl rendering. Research Report MPI-I-2000-4-001, Max-Planck-Institut für Informatik, Stuhlsatzenhausweg 85, 66123 Saarbrücken, Germany, February 2000.

[9] W. Heidrich, K. Daubert, J. Kautz, and H. -P. Seidel. Illuminating micro geometry based on precomputed visibility. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 455–464. ACM Press/Addison-Wesley Publishing Co., 2000.

[10] R. L. Cook. Shade trees. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pages 223–231. ACM Press, 1984.

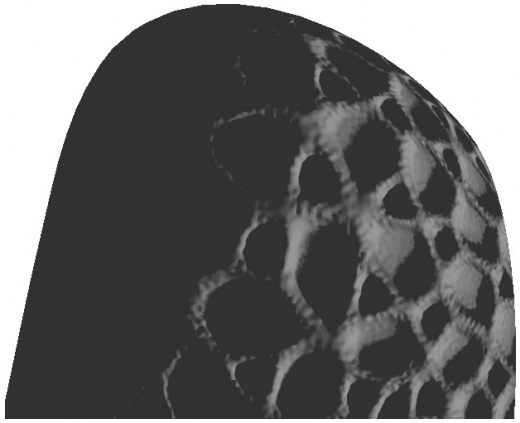


(a) bumpmap shadows by the previous method

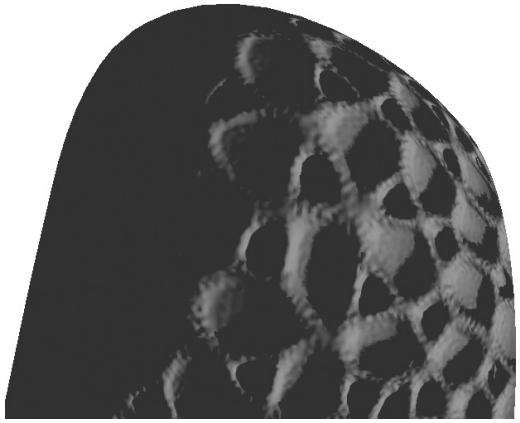


(b) bumpmap shadows by the proposed method

Figure 6. Bumpmap shadows on a Bezier surface.



(a) bumpmap shadows by the previous method



(b) bumpmap shadows by the proposed method

Figure 7. Bumpmap shadows on a Bezier surface (heights of the bump map are inverted).

[11] L. Williams. Casting curved shadows on curved surfaces. In *Proceedings of the 5th annual conference on Computer graphics and interactive techniques*, pages 270–274. ACM Press, 1978.

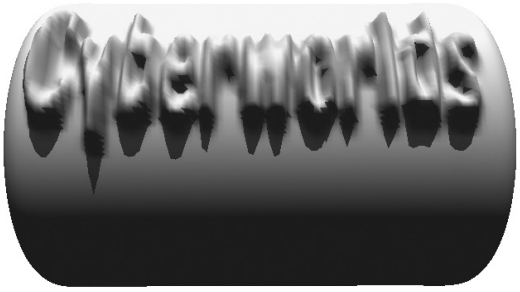
[12] W. T. Reeves, D. H. Salesin, and R. L. Cook. Rendering antialiased shadows with depth maps. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 283–291. ACM Press, 1987.

[13] F. C. Crow. Shadow algorithms for computer graphics. In *Proceedings of the 4th annual conference on Computer graphics and interactive techniques*, pages 242–248. ACM Press, 1977.

[14] L. Wang, X. Wang, X. Tong, S. Lin, S. Hu, B. Guo, and H. -Y. Shum. View-dependent displacement mapping. *ACM Transactions on Graphics (TOG)*, 22(3):334–339, 2003.

(a) bumpmap shadows rendered by using the previous method

(b) bumpmap shadows rendered by using the proposed method



(c) displacement mapping + shadow volume

Figure 8. Validation of the proposed method.