View-Dependent Displacement Mapping

Lifeng Wang Xi Wang[‡] Xin Tong Stephen Lin Shimin Hu[‡] Baining Guo Heung-Yeung Shum

Microsoft Research Asia[*] ‡Tsinghua University[†]

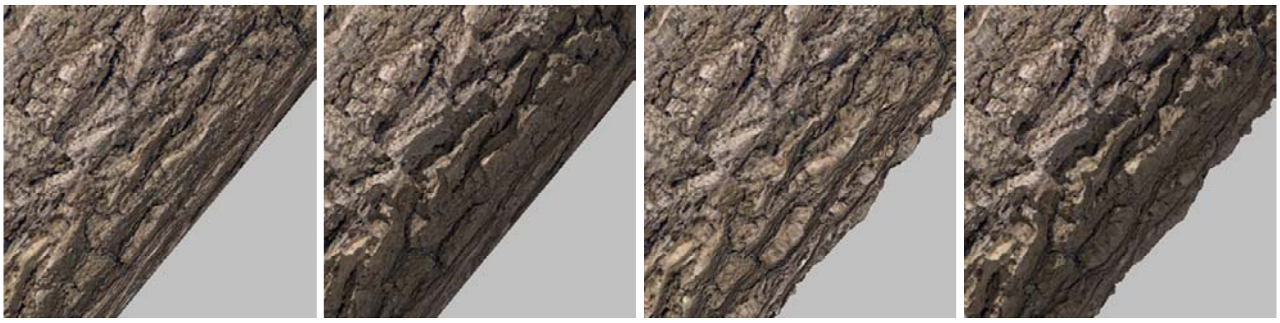

Figure 1: Comparison of different mesostructure rendering techniques: (a) bump mapping, (b) horizon mapping, (c) conventional displacement mapping, and (d) view-dependent displacement mapping with self-shadowing.

Abstract

Significant visual effects arise from surface mesostructure, such as fine-scale shadowing, occlusion and silhouettes. To efficiently render its detailed appearance, we introduce a technique called view-dependent displacement mapping (VDM) that models surface displacements along the viewing direction. Unlike traditional displacement mapping, VDM allows for efficient rendering of self-shadows, occlusions and silhouettes without increasing the complexity of the underlying surface mesh. VDM is based on per-pixel processing, and with hardware acceleration it can render mesostructure with rich visual appearance in real time.

CR Categories: I.3.3 [Computer Graphics]: Picture/Image Generation; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism.

Keywords: Reflectance and shading models, mesostructure, displacement maps, hardware rendering

1 Introduction

Accurate rendering of fine-scale geometric features and their associated visual effects yields vivid object appearance. This level of detail, commonly referred to as mesostructure [Koenderink and Doorn 1996; Dana et al. 1999], lies between that of meshes, which provide an efficient representation of gross shape, and bi-directional reflectance functions (BRDFs) that describe the light scattering effects of micro-scale material structure. The high-frequency visible geometry of mesostructure leads to complex visual effects including fine-scale shadowing, occlusion and silhouettes. These characteristics are necessary to include for realistic appearance, but are also difficult to render efficiently.

Mesostructures are typically rendered using techniques such as bump mapping, horizon mapping or displacement mapping. A comparison of different mesostructure rendering methods in handling visual effects is provided in Table 1. As displayed in Fig. 1, bump mapping [Blinn 1978] provides basic shading but not shadowing, occlusion, and silhouettes. Horizon mapping [Max 1988] improves realism by adding self-shadows, but occlusion and silhouettes are still missing. A combination of displacement mapping and horizon mapping could potentially yield most of the visual effects, but achieving real-time performance is challenging. Although conventional displacement mapping can be implemented in hardware, its performance for mesostructure rendering is limited by the large number of vertices that result from the considerable mesh subdivision required. Adding shadows by horizon mapping would further aggravate the performance problem.

Inspired by previous work on horizon mapping and bi-directional texture functions (BTFs), we introduce in this paper a pixel-based technique called view-dependent displacement mapping (VDM) for efficient rendering of mesostructures. Unlike the traditional displacement mapping [Cook 1984], VDM represents displacements along the viewing direction instead of the mesh normal direction. This view dependency allows VDM to interactively compute self-shadows as well as shading, occlusion and silhouettes. Most importantly, all these visual effects are achieved without increasing the complexity of the underlying surface mesh. Implementation of VDM can be done in current graphics hardware as per-pixel operations, bringing greater processing efficiency than methods that require fine mesh subdivision (e.g., [Cook et al. 1987]). The primary benefits of VDM are as follows:

- Renders major visual effects of mesostructure.
- Achieves high frame-rates with hardware acceleration of a per-pixel algorithm.

Interreflection is not directly included in VDM and is approximated

[*]3F Beijing Sigma Center, No 49 Zhichun Road, Haidian District, Beijing 100080, P R China, email: bainguo@microsoft.com

[†]This research was done when Xi Wang was working as a visiting student at Microsoft Research Asia.

in our implementation as an overall effect using an ambient illumination term.

The remainder of the paper is organized as follows. The subsequent section reviews previous work. In Section 3, we describe our mesostructure rendering algorithm and its hardware implementation. Section 4 presents results, and the paper concludes with future work in Section 5.

	Fine-scale Visual Effect			
	Shadow	Occlusion	Silhouette	Interreflection
Bump Mapping				
Horizon Mapping	X			
Displ. Mapping		X	X	
BTF	X	X		X
VDM	X	X	X	

Table 1: Comparison of mesostructure rendering methods. For bump mapping and displacement mapping, this table refers to their conventional algorithms.

2 Related Work

The simplest technique for mesostructure rendering is bump mapping [Blinn 1978], which perturbs mesh normals to match those of the fine geometric detail. Unfortunately, bump mapping does not handle other visual effects of mesostructure. To deal with self-shadows, Max introduced horizon mapping [Max 1988] as an extension to bump mapping. Sloan and Cohen demonstrated that horizon mapping can be done at an interactive rate with hardware acceleration [Sloan and Cohen 2000]. To account for occlusion, Becker and Max proposed redistribution bump mapping [Becker and Max 1993], where the bump map normals are adjusted according to the viewing direction. However, they also point out that problems arise when horizon mapping and redistribution bump mapping are put together. Heidrich et al. enhanced bump mapping using precomputed visibility [Heidrich et al. 2000]. Their technique simulates self-shadowing and interreflection but does not account for occlusions and silhouettes.

Traditionally, displacement mapping [Cook 1984] is implemented by subdividing the original geometry into a large number of micro polygons whose vertices can be displaced in the normal direction [Cook et al. 1987]. Adaptive remeshing methods [Gumhold and Hüttner 1999; Doggett and Hirche 2000] have been proposed to reduce the number of subdivisions; nevertheless, the number of generated triangles is still large and remains difficult to process in real time without development of hardware support.

To have the features of displacement mapping without the complexity of modifying original geometry, several hardware-based techniques have been based on image layers [Kautz and Seidel 2001; Meyer and Neyret 1998; Lensch et al. 2002]. In this 3D texture approach, each triangle is rendered several times according to the number of slice layers, and self-shadowing is not simulated. Several other methods avoid extensive subdivision by employing per-pixel processing. Most of these are based on ray-tracing [Patterson et al. 1991; Pharr and Hanrahan 1996; Smits et al. 2000], which is computationally slow. Schaufler et al. presented a backward warping algorithm that efficiently finds pixel appearance from many reference depth images [Schaufler and Priglinger 1999]; however, a hardware implementation does not exist for this complicated method. Another warping-based technique, the relief texture mapping by Oliveira et al. [Oliveira et al. 2000], captures silhouettes and occlusion, but self-shadowing and shading are absent.

An image-based approach to mesostructure rendering is to sample surface appearance under various lighting and viewing directions. The bi-directional texture function introduced by Dana et al. captures shading, self-shadowing, and occlusion [Dana et al. 1999]. The polynomial texture map (PTM) proposed by Malzbender et al. represents shading and self-shadowing but not occlusion [Malzbender et al. 2001]. Both BTF and PTM do not render silhouettes.

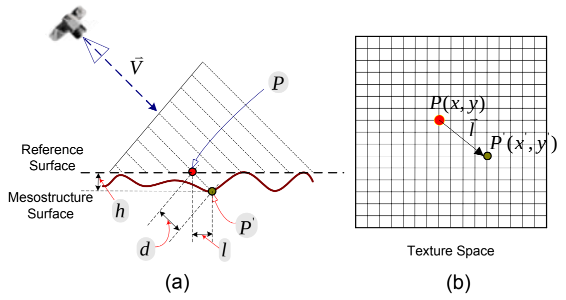

Figure 2: Mesostructure displacement with respect to view direction. (a) Displacement geometry. (b) Texture space offset.

3 VDM Rendering

Our mesostructure rendering technique first takes as input a textured height field sample and converts it to the VDM representation. This VDM data is rendered by a per-pixel algorithm that is implemented in hardware. Details of the VDM approach are described in this section.

3.1 VDM Definition

A view-dependent displacement map records the distance of each mesostructure point from the reference surface along the viewing direction, as illustrated in Fig. 2. For a given viewing direction $\mathbf{V} = (\theta, \phi)$ expressed in spherical coordinates, each reference surface point P with texture coordinate (x, y) projects to a mesostructure surface point P' that has texture coordinate (x', y'). For different viewing directions, P clearly would project onto different points on the mesostructure surface, so the displacement value d is dependent on x, y, θ, ϕ. Although other view-dependent parameters such as h or l could equivalently represent the displacement, d is less affected by sampling errors that would arise at viewing directions close to the surface normal or tangent.

The local curvature of the mesh surface also affects the projection point P' as shown in Fig. 3. When displacements for a flat reference surface as illustrated in Fig. 3(a) are mapped onto a curved surface as in Fig. 3(b), the displacements are no longer correct, and for some viewing directions such as $\mathbf{V}_{n-1}$ there does not even exist an intersection with the mesostructure surface. Curvature is generally ignored in other rendering algorithms such as bump mapping, horizon mapping, BTF and PTM. However, neglect of this factor can lead to inaccurate silhouettes and a warped appearance of the mesostructure.

With the consideration of curvature, view-dependent displacement information can be organized into a five-dimensional VDM function $d_{VDM}(x, y, \theta, \phi, c)$, where x, y are the texture coordinates on the reference surface, θ, ϕ are the spherical angles of the viewing direction in the local coordinate frame on the reference surface, and c is the reference surface curvature along the viewing direction. If an intersection with the mesostructure exists for (x, y) in direction (θ, ϕ) with curvature c, then the VDM function takes the value of the view-dependent displacement. If the intersection does not exist, then d_{VDM} is set to -1. In our implementation, the mesostructure surface is defined inwards of the reference surface, so negative displacements are invalid.

The VDM function is built by first forming a fine mesh for a mesostructure with zero curvature, and then using a ray-casting algorithm to calculate the view-dependent displacement values. An example of a VDM function computed from a height field is shown in Fig. 4.

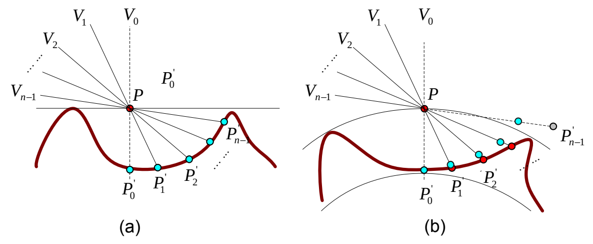

Figure 3: View-dependent displacement variation for curved surfaces.

While it is possible to compute the full 5D VDM by forming a mesostructure mesh for various curvature values, we in practice take a less computationally expensive approach that is based on the zero-curvature mesostructure mesh. Details of this method are presented in the Appendix.

3.2 Rendering Algorithm

After computing a VDM for a given mesostructure surface, it can be mapped to a new surface as a high-dimensional texture. This mapping should result in texture coordinates on the object surface that have locally uniform scale and little distortion, with the mesostructure pattern closely maintained. Since the mesostructure is a 3D volume, it is also requisite that its dimensions be scaled equally in the mapping.

To render this mapping, the normals of the object surface must first be calculated from the height field. Each surface normal and the two orthogonal directions of the texture define a local coordinate system in which the viewing and lighting directions are expressed.

The parameters of each pixel on the rasterized surface are then computed. These parameters consist of the texture coordinate $T = (x, y)$, the illumination direction $\mathbf{L} = (\theta_L, \phi_L)$, the viewing direction $\mathbf{V} = (\theta_V, \phi_V)$, and the local curvatures c_V, c_L along $\mathbf{V}$ and $\mathbf{L}$ respectively. From the two principal curvature directions C_{max}, C_{min} and their corresponding curvature values c_{max}, c_{min}, the curvature along the viewing direction $\mathbf{V}$ can be computed as

$$c_V = \frac{c_{max}(\mathbf{V} \cdot C_{max})^2 + c_{min}(\mathbf{V} \cdot C_{min})^2}{1 - (\mathbf{V} \cdot \mathbf{N})^2}.$$

The curvature c_L along the lighting direction $\mathbf{L}$ is computed similarly. Since the cost of calculating the parameter values for each pixel is unacceptably large, in practice we calculate the parameters per vertex and then interpolate them for each pixel.

After calculating these pixel quantities, VDM rendering of detailed geometry proceeds with the following four steps, organized in the flowchart of Fig. 6.

Silhouette Determination: The VDM function includes an explicit representation of point visibility along the mesostructure silhouette. When $d_{VDM} = -1$, then the corresponding line of sight intersects no detailed geometry. In this case, the pixel need not be processed and the rest of the algorithm is skipped. In practice, for each pixel we use its nearest sample value in texture space for silhouette determination.

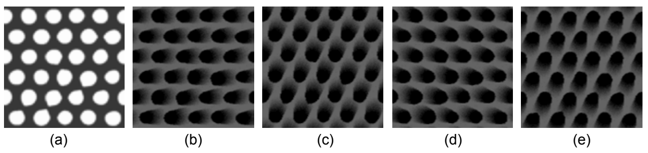

Figure 4: VDM images synthesized from a mesostructure height field. (a) Height field, (b-e) VDM images for different viewing directions, where the gray-level is proportional to the view-dependent displacement from the bounding reference surface.

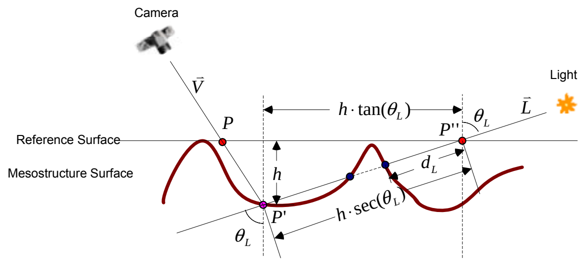

Figure 5: Shadow determination in VDM

Real Texture Coordinate Calculation: Since properties of the detailed geometry are parameterized by texture coordinates on the planar reference surface, the actual texture coordinate of each mesostructure intersection point needs to be determined.

For a planar reference surface, the offset between the real texture coordinate T' and original texture coordinate $T = (x, y)$ can be computed as $dT = d_{VDM}(x, y, \theta_V, \phi_V, c_V)V_{xy}$, where $V_{xy} = (sin\theta_V cos\phi_V, sin\theta_V sin\phi_V)$ is the projection of the viewing vector onto the local tangent plane. For a curved reference surface, the texture offset can be computed in the same way from the extended VDM displacement for curved surfaces described in the Appendix.

The real texture coordinate T' is then obtained by adding this offset to the original texture coordinate T, i.e., $T' = T + dT$.

Shadow Determination: Shadowing of the intersection point from the light source can be determined from the VDM function. We first describe this computation for a flat surface as illustrated in Fig. 5. Let P be the mesh surface point and P' be the intersection point on the detailed geometry. Let h denote the distance from P' to the surface as given by the mesostructure height field. The point P'' represents the reference surface intersection of $\mathbf{L}$ passing through P'.

Since the surface is flat, the texture coordinate of P'' can be geometrically determined as $T = T + h \cdot tan(\theta_L) \cdot \mathbf{L}$. The view-dependent displacement of $P'' = (x'', y'')$ along $\mathbf{L}$ is given by $d_L = d_{VDM}(x'', y'', \theta_L, \phi_L, 0)$.

The distance $h \cdot sec(\theta_L)$ between P' and P'' is compared to d_L to determine the presence of shadow. When the two quantities are equal, the light source is not occluded from P'; otherwise, P' is in shadow.

For a curved object surface, the computation of T requires solving some trigonometric equations. To reduce run-time computation, a table indexed by the variables h, θ_L and c_L could be built. We have found, however, that the effect of curvature on shadow determination is generally unnoticeable, so it is ignored in our implementation.

Pixel Shading: If the pixel is not in shadow, a pre-defined reflectance model is used to compute pixel appearance. Otherwise, its appearance is computed for ambient illumination only.

3.3 Data Decomposition and Compression

The rapid development of programmable graphics hardware provides opportunities for real-time implementations of our VDM rendering algorithm. However, some hardware constraints should be overcome. First, the maximum dimensionality of texture maps in current hardware poses a problem for high-dimensional maps such as VDM. Second, memory constraints present an obstacle to the substantial amount of VDM data. Throughout this paper, we use mesostructure height fields of resolution 128×128 with 32×8 viewing directions and 16 curvatures. The corresponding 64 MB of VDM data consumes most of the graphics hardware memory and cannot efficiently be loaded.

To deal with these restrictions, we decompose and compress the data by singular-value decomposition (SVD). SVD has the benefit

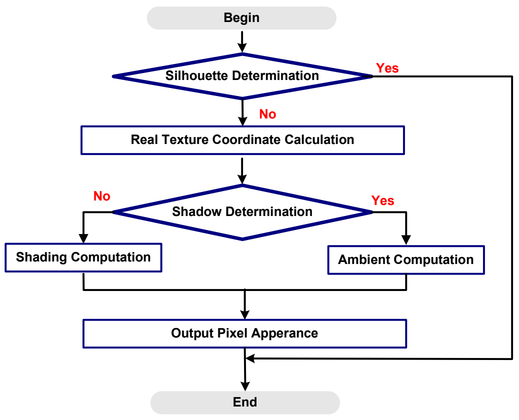

Figure 6: Flowchart of per-pixel VDM rendering.

that it can efficiently decompose a high-dimensional map into two lower-dimensional maps. Moreover, reconstruction is performed using only linear arithmetic operations, which can easily be handled in the hardware. For data with high correlation, a small number of eigen-functions is sufficient for good reconstruction.

In VDM, when a ray along a view direction has no intersection with the mesostructure, it is labelled with the special value -1. When the silhouette exhibits high frequency, the number of eigen functions needed for acceptable reconstruction is dramatically increased, which leads to difficulties in hardware implementation.

To address this problem, we employ an auxiliary 4D Maximal View polar angle Map (MVM). For a densely sampled VDM, the MVM $\theta_{MVM}(x, y, \phi, c) \equiv max\ \theta$ such that

$$d_{VDM}(x, y, \theta, \phi, c) \neq -1,$$

where x, y are texture coordinates, ϕ is the azimuth angle of the viewing direction, and c is the local curvature. After this computation, the -1 value of the corresponding polar viewing angle in the original VDM is then replaced with this maximum polar viewing angle. For our mesostructure samples, the MVM size is 4 MB, and combined with the VDM data, the total data size becomes 68 MB.

The high-dimensional VDM and MVM are then reorganized into a 2D matrix

$$A = [A_{VDM}, A_{MVM}]$$

such that the rows are indexed by x, y, ϕ and the columns are indexed by θ and/or c. Although these maps could be differently organized for decomposition, experiments have shown that our solution provides the best compromise between accuracy and storage for all data used in this paper.

Applying SVD to A gives $A = U\lambda E^T = WE^T$, where $E = [E_{VDM}, E_{MVM}]$ contains the eigen functions of A and $W = U\lambda$ contains the weights of the eigen functions. From this, the two maps can be expressed as

$$d_{VDM}(x, y, \theta, \phi, c) = \sum_i W^i(x, y, \phi)E^i_{VDM}(\theta, c)$$

$$\theta_{MVM}(x, y, \phi, c) = \sum_i W^i(x, y, \phi)E^i_{MVM}(c)$$

where i indexes the eigen functions in E and the eigen function weights in W. Note that a common weight function is used for the two maps to reduce storage.

Since the eigenvalues decrease rapidly in magnitude, good rendering accuracy can be achieved with a small number of eigen functions. For the VDM data used in this paper, we employ only 8 VDM and 4 MVM eigen functions for rendering. By this decomposition and compression, the original 68 MB of VDM is reduced to 4 MB.

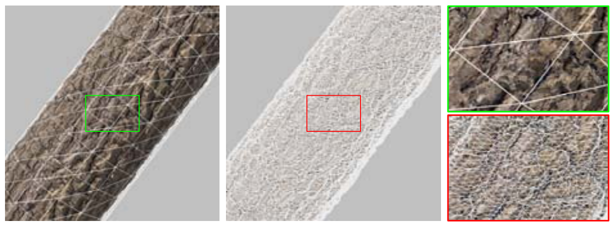

Figure 7: Mesh complexity in VDM and displacement mapping for rendering similar level of detail images. (a) VDM mesh, (b) Displacement map mesh. (c) Zoomed views of the green and red boxes in (a) and (b) respectively.

4 Results

Our system is implemented on a 1.4 GHz 768MB Pentium IV PC with an ATI Radeon 9700 Pro 128MB graphics card. The hardware-accelerated VDM rendering is implemented as a single rendering pass using Pixel Shader 2.0 with OpenGL. The complexity of the pixel shader is 40 arithmetic instructions and 14 texture lookups. All VDM data in this paper is of resolution 128x128 and sampled for 32x8 viewing directions and 16 curvatures between -2.0 and 3.0, which are interpolated for intermediate values. The number of sampled values can be increased for greater accuracy at the expense of data size. Additionally, VDM as a texture map is easily anti-aliased by mip-mapping all data dimensions.

Important visual effects such as shadowing, occlusion, and silhouettes are all captured well by the VDM technique. Fig. 1 compares VDM with hardware-accelerated bump mapping, horizon mapping and conventional displacement mapping where the geometry is subdivided and displaced before software rendering.[1] The added mesh complexity of displacement mapping as exemplified in Fig. 7 prevents rendering in real time.

Overall, the VDM algorithm has a fill rate performance of approximately 50M/sec and a per-triangle processing speed of about 15M/sec. A main reason for the real-time performance of VDM is that it is based on per-pixel processing. Additional examples of VDM rendering are displayed in Fig. 9. These models have a screen size of 512×512 and rendering performance listed in the figure.

Renderings under different levels of data compression are displayed in Fig 8. It can be seen that even with only a small number of eigen functions, the mesostructure maintains an appearance close to that for the original uncompressed VDM.

5 Conclusion

In this paper, we presented a pixel-based method for real-time rendering of surface mesostructure and its associated visual effects, including shadowing, occlusion, and silhouettes. A number of research topics remain to be explored. We plan to investigate methods for efficient capture of VDM functions from real world surfaces. Also, VDMs will be extended to model open surface boundaries, which are not handled in our current implementation. Another interesting topic is to incorporate interreflections and support spatially-variant BRDFs in the VDM framework.

Acknowledgments: We would like to thank the anonymous reviewers for their constructive critiques. Many thanks to Jeff Royle

[1]Currently, adaptive tessellated displacement mapping is supported only by the Matrox Parhelia, but information such as source code, API support and a proper driver are not available. ATI claims to support pre-sampled displacement mapping on the Radeon9700. However, our personal communications with ATI technical staff revealed that no proper driver is available now to access this feature.

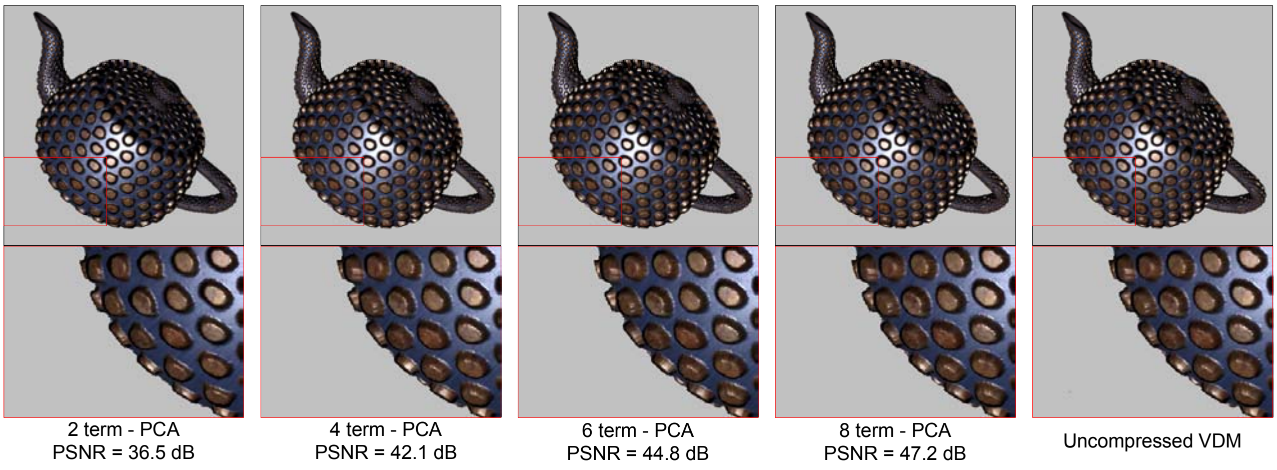

Figure 8: VDM rendering under different levels of compression.

of ATI Technologies Inc., Dave McCoy and Chas Boyd of Microsoft Corp. for useful discussion about the latest status of hardware displacement mapping. Prof. Shimin Hu was supported by NSFC (No. 60225016) and National Basic Research Project of China (No. 2002CB312102)

References

BECKER, B. G., AND MAX, N. L. 1993. Smooth transitions between bump rendering algorithms. *Computer Graphics (SIGGRAPH '93 Proceedings)*, 183–190.

BLINN, J. F. 1978. Simulation of wrinkled surfaces. *Computer Graphics (SIGGRAPH '78 Proceedings) 12*, 3, 286–292.

COOK, R. L., CARPENTER, L., AND CATMULL, E. 1987. The reyes image rendering architecture. *Computer Graphics (SIGGRAPH '87 Proceedings)*, 95–102.

COOK, R. L. 1984. Shade trees. *Computer Graphics (SIGGRAPH '84 Proceedings) 18*, 3, 223–231.

DANA, K. J., NAYAR, S. K., VAN GINNEKEN, B., AND KOENDERINK, J. J. 1999. Reflectance and texture of real-world surfaces. *ACM Transactions on Graphics 18*, 1, 1–34.

DOGGETT, M., AND HIRCHE, J. 2000. Adaptive view dependent tessellation of displacement maps. *Eurographics Workshop on Graphics Hardware*, 59–66.

GUMHOLD, S., AND HÜTTNERT, T. 1999. Multiresolution rendering with displacement mapping. *Eurographics Workshop on Graphics Hardware*, 55–66.

HEIDRICH, W., DAUBERT, K., KAUTZ, J., AND SEIDEL, H.-P. 2000. Illuminating micro geometry based on precomputed visibility. *Computer Graphics (SIGGRAPH '00 Proceedings)*, 455–464.

KAUTZ, J., AND SEIDEL, H.-P. 2001. Hardware accelerated displacement mapping for image based rendering. *Graphics Interface*, 61–70.

KOENDERINK, J. J., AND DOORN, A. J. V. 1996. Illuminance texture due to surface mesostructure. *Journal of the Optical Society of America 13*, 3, 452–463.

LENSCH, H. P. A., DAUBERT, K., AND SEIDEL, H.-P. 2002. Interactive semi-transparent volumetric textures. *Proc. Vision, Modeling and Visualization*, 505–512.

MALZBENDER, T., GELB, D., AND WOLTERS, H. 2001. Polynomial texture maps. *Computer Graphics (SIGGRAPH '01 Proceedings)* (August).

MAX, N. 1988. Horizon mapping: shadows for bump-mapped surfaces. *The Visual Computer 4*, 2, 109–117.

MEYER, A., AND NEYRET, F. 1998. Interactive volumetric textures. *Eurographics Workshop on Rendering*, 157–168.

OLIVEIRA, M. M., BISHOP, G., AND MCALLISTER, D. 2000. Relief texture mapping. *Computer Graphics (SIGGRAPH '00 Proceedings)*, 359–368.

PATTERSON, J. W., HOGGAR, S. G., AND LOGIE, J. R. 1991. Inverse displacement mapping. *Computer Graphics Forum 10*, 2, 129–139.

PHARR, M., AND HANRAHAN, P. 1996. Geometry caching for ray-tracing displacement maps. *Eurographics Workshop on Rendering*, 31–40.

SCHAUFLER, G., AND PRIGLINGER, M. 1999. Efficient displacement mapping by image warping. *Eurographics Workshop on Rendering*, 175–186.

SLOAN, P.-P., AND COHEN, M. F. 2000. Interactive horizon mapping. *Eurographics Workshop on Rendering* (June), 281–286.

SMITS, B., SHIRLEY, P., AND STARK, M. M. 2000. Direct ray tracing of displacement mapped triangles. *Eurographics Workshop on Rendering*, 307–318.

Appendix

For ease of explanation, let us consider a given point (x_0, y_0) and azimuth angle ϕ_0 of the viewing direction. Suppose we have n sampled viewing elevation angles $(\theta_0, \theta_1, ..., \theta_{n-1})$ uniformly distributed over $[0, \pi/2]$ and intersecting the microgeometry at points $P_0, P_1, ..., P_{n-1}$, as shown in Fig. 3(a). The displacements of these points are used to compute the zero curvature values of the VDM, and can also be used for calculations of other curvatures.

Bending the zero-curvature microgeometry surface to have curvature $C = C_0$ causes the zero-curvature intersection points to correspond to new viewing angles $(\theta'_0, \theta'_1, ..., \theta'_{n-1})$ that are no longer a uniform sampling. To obtain the VDM values, we resample these non-uniform samples by linear interpolation.

In the resampling process, a couple special cases require attention. When the curvature is positive, some view-dependent displacement values may become invalid because no intersection with the mesostructure surface exists. In this instance, the VDM is assigned a value of -1. Another special case occurs when the sampled angles become reordered after resampling. This happens when a zero-curvature intersection point becomes occluded after bending. For this case, the occluded direction is discarded in the resampling operation.

To efficiently compute the texture offset for a curved surface at rendering time, we modify the VDM definition to

$$d_{VDM}(x, y, \theta_V, \phi_V, c_V) = \frac{l}{sin(\theta_V)},$$

where l is the distance between P and P' in texture space and can be geometrically computed from the triangle defined by P, P' and the center of curvature. This extended definition simplifies texture offset computation for curved surfaces as described in Section 3.1.

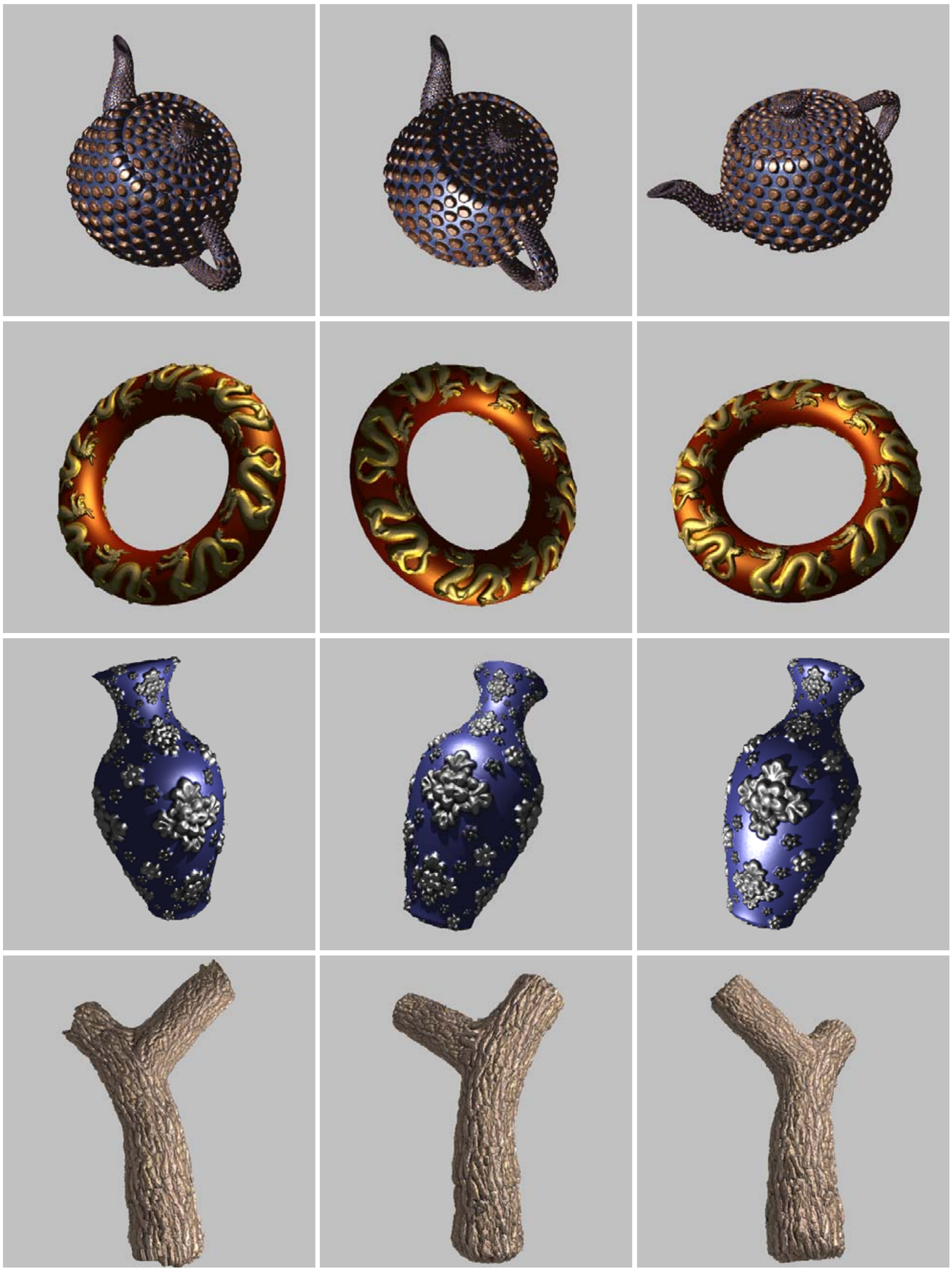

Figure 9: Examples of VDM rendering under different lighting and viewing directions. The number of triangles and rendering speed for each object are as follows. Teapot: 4032 tri., 104 FPS; Torus: 4800 tri., 112 FPS; Vase: 8640 tri., 130 FPS; Branch: 5888 tri., 139 FPS.