Real-time per-pixel rendering of textiles for virtual textile catalogues

Andy Spence[1], **Mike Robb**[1], **Mark Timmins**[2] **& Mike Chantler**[1]

[1]School of Mathematical & Computer Sciences, Heriot-Watt University, Edinburgh, EH14 4AS, U.K.
[2]School of Textiles & Design, Heriot-Watt University, Galashiels, Selkirkshire, 3HF TD1, U.K.

Keywords *retailing automation, 3D simulation, E-commerce, augmented reality.*

Abstract

We present recent results from an EPSRC funded project VirTex (Virtual Textile Catalogues). The goal of this project is to develop graphics and image-processing software for the capture, storage, search, retrieval and visualisation of 3D textile samples. The ultimate objective is to develop a web-based application that allows the user to search a database for suitable textiles and to visualize selected samples using real-time photorealistic 3D animation.

The main novelty of this work is in the combined use of photometric stereo and real-time per-pixel-rendering for the capture and visualisation of textile samples. Photometric stereo is a simple method that allows both the bump map and the colour map of a surface texture to be captured digitally. It uses a single fixed camera to obtain three images under three different illumination conditions. The colour map is the image that would be obtained under diffuse lighting. The bump map describes the small undulations of the surface relief. When imported into a standard graphics program these images can be used to texture 3D models. The appearance is particularly photorealistic, especially under changing illumination and viewpoints. The viewer can manipulate both viewpoint and lighting to gain a deeper perception of the properties of the textile sample. In addition, these images can be used with 3D models of products to provide extremely accurate visualisations for the customer.

Until recently these images could only be rendered using ray-tracing software. However, recent consumer-level graphics cards from companies such as Nvidia, ATI and 3Dlabs now provide real-time per-pixel shading. We have developed software that takes advantage of the advanced rendering features of these cards to render images in real-time. It uses photometrically acquired bump and colour maps of textiles to provide real-time visualisation of a textile sample, under user-controlled illumination, pose and flex.

1. Introduction

The photorealism of a computer-generated 3D scene illuminated by light sources can be enhanced in various ways. At a simple yet effective level a technique known as texture mapping is commonly used to this end. This involves 'pasting' a digital 2D image onto a 3D object composed of polygons to give it the appearance of having a textured surface (Figure 1b). Photorealism is certainly enhanced by doing so but an object rendered in this manner appears unnaturally smooth. Furthermore the texture itself will be unreactive to changing illumination conditions. These drawbacks limit the usefulness of texture mapping in e-commerce applications for the textile industry. The objective of this EPSRC project VirTex (Virtual Textile Catalogues) is to address and solve these issues with a view to achieving high-fidelity photorealistic 3D animations .

The word 'texture' is of course usually associated with rough or bumpy surfaces in the real world. The appearance of such surfaces can change dramatically when illumination conditions are altered (Figure 2). It is important to model this effect if scene photorealism is to be enhanced. This is especially true for animations in which a textured object is moving relative to the light source. Modelling the effect is essentially what is achieved by relighting techniques whereby a texture is reproduced under user-specified illumination conditions using data derived from multiple images of the texture under varying illumination. Not all relighting methods are suitable in this case, however. For example, Malzbender [6] introduced polynomial texture maps as an effective way to model luminance but it is the *scene* which is reconstructed. This method therefore does not lend itself to mapping a texture onto a 3D object.

Instead it is useful to think of a geometric object and its texture as separate entities as with texture mapping. In this case 'bump mapping' which was introduced by Blinn [1] is the appropriate technique, however. A bump map is used to store information about the topography of a texture in terms of its surface normals. Since normals are a key element in lighting calculations this technique actually allows the surface of an object to both appear rough and also be reactive to changing illumination conditions (Figure 1c,d). Importantly this is achieved without an increase in the geometric complexity

of the object itself. The fact that both the bump map and its integrated form, the displacement map (Figure 3), are universally used in computer graphics applications is also noteworthy.

In addition to the bump map, information pertaining to the colour of the texture is also required. This albedo map, technically defined as the ratio of reflected light to that incident on the surface, must also be determined. It can then used as a texture map in the rendering process to achieve a high degree of photorealism.

Whilst it is possible to utilise both the bump map and the albedo texture map in standard 3D packages, rendering is not carried out in real-time. However, for interactive applications real-time rendering is a must. Recent consumer-level graphics cards from companies such as Nvidia, ATI and 3Dlabs now provide real-time per-pixel shading. The advanced rendering features of these cards allows software to be developed which uses the photometrically acquired bump and albedo maps to provide real-time visualisation under user-controlled illumination, pose and flex.

In Section 2 the photometric stereo method which is used to capture the data required to generate the bump map and albedo map is described. How these maps which define the texture are utilised for real-time visualisation is then considered in Section 3. Conclusions regarding the whole process are finally drawn in Section 4.

2. Determining the Bump Map and the Colour Map of a Textile Texture

Photometric stereo (PS) is a classic computer vision technique for shape estimation which has been extensively researched over many years and which has found applications in diverse areas such as surface inspection, classification and recognition. Woodham's [2] original algorithm is based on reflectance maps which were introduced by Horn [3]. Significantly, a reflectance map links the topgraphy of a textured surface to the intensity in its corresponding image. Woodham demonstrated that three images of a surface under different illumination conditions are sufficient to uniquely determine both the surface orientation and the albedo or colour map – from the intersection of the three reflectance maps. Since this method presents a relatively simple way of obtaining the bump maps and albedo maps of textile samples it has therefore been utilised in our work for the VirTex project.

Over the years the PS algorithm has been refined and modified to cope with less than ideal conditions such as when shadows, specularities or interreflections are present [8,9,10,11,12,13]. The three image algorithm is still sufficient, however, to recover the surface normals and albedo for a diffuse surface with no shadows. In this case Lambert's Law applies such that the intensity value of a pixel is proportional to the cosine of the angle between the illumination vector, **l**, and the surface normal, **n**, of the corresponding surface facet scaled by the texture albedo, ρ. This is written in terms of a dot product in equation 1.

$$i(x,y) = \rho(\mathbf{l} \bullet \mathbf{n}) \qquad (1)$$

The direction of the illumination vector which points towards the surface facet is limited to that within a hemisphere above the facet. It is therefore intuitive to define it in terms of polar coordinates using the tilt angle, τ, and the slant angle, σ. These parameters are equivalent to the angles of latitude and longitude respectively and can be measured.

$$\mathbf{l} = (l_x, l_y, l_z)^T = (\cos\tau \sin\sigma, \sin\tau \sin\sigma, \cos\sigma)^T \quad (2)$$

Three images are captured under different illumination conditions in the PS algorithm and thus provide three simultaneous equations which can be written in the following form.

$$\begin{bmatrix} i_1 \\ i_2 \\ i_3 \end{bmatrix} = \rho \begin{bmatrix} l_{1,x} & l_{1,y} & l_{1,z} \\ l_{2,x} & l_{2,y} & l_{2,z} \\ l_{3,x} & l_{3,y} & l_{3,z} \end{bmatrix} \mathbf{n} \qquad (3a)$$

or equivalently

$$\mathbf{i} = \rho \mathbf{L} \mathbf{n} \qquad (3b)$$

It is therefore apparent that if both the intensity and illumination vectors have been measured, then the unknowns can be determined by inverting the illumination matrix, **L**.

$$\mathbf{t} = \mathbf{L}^{-1}\mathbf{i} \qquad (4)$$

where **t** = ρ**n**, and is the *scaled* surface normal.

Rather than store the three elements of $\mathbf{t} = (t_1, t_2, t_3)^T$, it is preferable to convert this data in order to separate the albedo from the surface normal. However, this does not imply the use of an extra storage variable. This is because the surface normal **n** can be written in terms of two variables, p and q, which are the partial derivatives of the surface.

$$\mathbf{n} = \frac{(-p,-q,1)^T}{\sqrt{p^2+q^2+1}} \qquad (5) \qquad \text{where } p = \frac{dz}{dx} \qquad q = \frac{dz}{dy} \qquad (6a,b)$$

These partial derivatives and the albedo are calculated from the scaled surface normal **t** at each pixel position as follows.

$$p = -\frac{t_1}{t_3} \qquad q = -\frac{t_2}{t_3} \qquad \rho = \sqrt{t_1^2 + t_2^2 + t_3^2} \qquad (7a,b,c)$$

The results are stored as images : the p & q images define the bump map for the texture whilst its corresponding colour map is the albedo image. For real-time per-pixel rendering, it is these images which are used as input for our custom 3D application (Figure 4).

It is noted that a standard consumer 3D application was used initially in our work (Figure 1). In this case the required input was a displacement map (Figure 3) rather than a bump map. This was generated by a frequency domain integration of the partial derivatives to give the height. The method used in our work is similar to that presented by Frankot & Chellappa [29].

3. Real Time Visualisation

The rendering of knit-wear and cloth materials has been researched by several authors [30,31,32,33]. Whilst these methods make use of 3D rendering techniques and achieve excellent photorealistic quality, they do not run in real-time. With the lumi-slice technique, a single frame can take anywhere from 15 to 30 minutes to be rendered. Another disadvantage is that this method requires the user to specify the weave pattern used to construct the material.

In order to allow rendering in real-time there are several tasks required to visualise the bump map and colour map images produced by the photometric stereo algorithm described in Section 2. These include the conversion of the image data into the internal data format used by the graphics accelerator, the configuration of each stage of the programmable graphics pipeline, and the transmission of geometry through the graphics pipeline.

3.1. Configuration of the graphics pipeline

Before loading the data in the graphics accelerator we need to use the pre-processing algorithm to calculate the outward normals used to define the bump map. This section describes how we use this data to implement per-pixel bump-mapping in hardware. Implementation of the graphics pipeline requires four separate stages to be designed:

1. The pre-processing stage
2. The vertex transformation stage
3. The per-pixel lighting stage
4. The rendering stage

3.1.1. Design of the pre-processing stage

Using the photometric stereo method, the initial texture data is in the form of a set of monochrome images in floating-point format. The first image defines the albedo colour map. The other two images define the P and Q partial derivatives of the surface with respect to each axis. The first image is converted for use by the graphics hardware simply by scaling and converting the image data from 32-bit floating point down to 8-bits. Conversion of the two gradient images is achieved in the following way.

Given the values of gradient it is possible to calculate the equivalent angle in radians for each pixel:

$$\phi_p = \tan^{-1}(p) = \tan^{-1}\left(\frac{dz}{dx}\right) \qquad \phi_q = \tan^{-1}(q) = \tan^{-1}\left(\frac{dz}{dy}\right) \qquad (8)$$

From these two angles, it is possible to calculate the partial derivatives of the surface, and the normal vector from the normalized cross product of the two values:

$$\mathbf{p}' = (\cos(\phi_{p)}, 0, \sin(\phi_p))^T \qquad \mathbf{q}' = (0, \cos(\phi_q), \sin(\phi_q))^T \qquad \mathbf{n} = |\mathbf{p}' \times \mathbf{q}'| \tag{9}$$

It should be noted that while an equation used to derive the outward normal **n** has been given in (5), this value is not actually calculated until required for rendering. The main reason for doing this is to save on storage space and transmission bandwidth for networked applications. The resulting vector is then scaled and biased for compression into an 8-bit signed RGB colour value. However, with future graphics accelerators such as the Nvidia's GeForce FX, it will be possible to use the floating-point data directly.

3.1.2 Vertex transformation

The transformation of vertex information is implemented using a vertex program, as this is the most efficient way of implementing the algorithm. A detailed explanation of this algorithm is described in "Efficient Bump Mapping Hardware" [19].

Rendering a bump-mapped model requires that the two additional direction vectors (tangent normal and binormal) specifying the local tangent space for the vertex are sent along with the outward normal, vertex and texture coordinates

For each vertex, the location and direction of the current light source is transformed into tangent space. This allows the half-angle vector between the eye vector and the outward normal to be calculated. This vector must also be normalised before being used in the lighting equation. This is achieved by using two texture cube maps to implement vector normalisation. As per-pixel lighting is required, the lighting equation is implemented in the register combiner stage.

To allow the texture coordinates to match the scale of a 3D model, two texture matrices are used to transform the texture coordinates prior to rendering.

3.1.3 Per-pixel lighting

Per-pixel lighting of the graphics pipeline is implemented using the register combiner unit of the graphics chip. It is noted that not all consumer-level graphics cards are programmable in this way; Nvidia's Geforce4 Ti4600 has this feature, however, and it is this card which has been used in our work. It allows the lighting model to incorporates ambient, diffuse and specular components on a per-pixel basis.

At this point, the following texture data is available:

1. Pixel colour of the base texture
2. Light source direction normal
3. Half-angle normal
4. Encoded 8-bit RGB normal from the bump map texture

The register combiner units are used in the following way. The first register unit is always used to calculate the dot products between the light-source direction normal and, the half-angle and bump map normal. The second register combiner is always used to implement the diffuse colour calculation. The remaining six register combiner units are used as required to raise the dot product result to the corresponding specular power. The final register combiner unit is used to combine the ambient, diffuse and specular lighting values together.

3.1.4 Rendering stage

For this project, Bezier patches were chosen as the basic geometric primitive. There were two reasons for this decision. The first reason was that animation and CAD users have used these in the past. The other reason is that the evaluation stage of this primitive can easily be modified to calculate the required tangent space coordinates.

The base texture and bump map texture are loaded into the graphics accelerator texture memory as 8-bit RGBA textures. The alpha channel of the base texture is used to define a transparency map, while the alpha channel of the bump map texture is used to modulate the specular term of the lighting model, and so define a gloss map. In order to implement trimmed surfaces, the regions of the patch that are removed are made invisible by setting the alpha channel to zero. Two other texture units are used to implement the normalisation stage using an environment cube-map.

One of two vertex programs is used to render the patch using either a directional or local point light source.

Each patch of the geometric model is evaluated in software and sent to the graphics accelerator.

3.1.5 Human-Computer Interaction

To allow the user to view the model with as much freedom as possible, the user interface has been designed to allow the user to control the position of the model, light sources and camera

independently. The operations supported include rotating the model, rotating and zooming both the camera and light-sources. All objects can be allowed to rotate automatically, to brake automatically, or to only rotate whenever the user moves the mouse. Light sources are rendered as a sphere in order to give the user feedback as to where the light source is located and moving.

3.1.6 Applications of photometric texture acquisition with 3D graphics hardware

To demonstrate how this technique can be used with existing 3D graphics hardware, we have taken the original Utah Teapot and applied a photometrically acquired bump-mapped texture of a knitted textile onto each Bezier patch. The final model is then rendered in real-time using OpenGL. Movement of the light source demonstrates how the bump-mapped texture can be used to generate photo-realistic images. It should be noted that while bump-mapping greatly improves the detail of the surface facing the observer, giving the illusion of a rough surface, when the same surfaces are viewed edge on, it is obvious that the surface is smooth. This is demonstrated in figure 4.

4. Conclusions

In this paper, we present a method of photometrically acquiring the image of a textile in terms of a bump map and a texture map. We have used the acquired data to implement real-time rendering of textured 3D models. With high performance graphics capability rapidly becoming available on consumer level hardware, this technique can easily be used to create virtual catalogues accessible by standard web browsers. This technology could also be applied to CAD and CAGM to allow designers to rapidly prototype designs in virtual reality.

5. References

1. Blinn, J. F. , 1978, “Simulation of Wrinkled Surfaces”, ACM Special Interest Group on Computer Graphics and Interactive Techniques, pp. 286-292. Also in Tutorial : Computer Graphics : Image Synthesis 307 – 313.

2. Woodham, R. J., 1980, “Photometric method for determining surface orientation from multiple images” Optical Engineering, 19(1),139 – 144.

3. Horn, B., 1986, “Robot Vision”, MIT Press.

4. Barsky, S. & Petrou, M., 2001, “Colour photometric stereo : simultaneous reconstruction of local gradient and colour of rough textured surfaces”

5. G. McGunnigle,1998, “The classification of texture surfaces under varying illumination direction”, PhD thesis, Dept. of Computing and Electrical Engineering, Heriot-Watt University.

6. Malzbender, T., Gelb D., Wolters, H., 2001, “Polynomial Texture Maps”, Siggraph, 519 –528.

7. Spence, A. D., 2003, “Optimal Illumination for Three-Image Photometric Stereo”, Research Memo CS2003/02, School of Mathematics & Computer Science, Heriot-Watt University.

8. Coleman, E. N., Jain, R., 1982, “Obtaining 3-Dimensional Shape of Textured and Specular Surfaces using Four-Source Photometry”, Computer Graphics and Image Processing, 18, 309-328

9. Rushmeier, H., Taubin, G., Gueziec, A., 1997, “Applying shape from lighting variation to bump map capture”, Eurographics Rendering Workshop Proceedings ‘97, 35 –44

10. Lee, K.M., Kuo, C.C.J., 1992 “Shape Reconstruction from Photometric Stereo”, Computer Vision and Pattern Recognition '92, Illinois.

11. Yamada, T., Saito, H., Ozawa, S., 1998, “3D Reconstruction of Skin Surface from Image Sequence”, Proc. Of MVA98: Workshop of Machine Vision Applications, 384-387.

12. Hansson, P., Johansson, P., 2000, “Topography and reflectance analysis of paper surfaces using a photometric stereo method”, Optical Engineering, 39(9), 2555-2561.

13. Schluns, K., 1997, “Shading Based 3D Shape Recovery in the Presence of Shadows”, Proc. First Joint Australia & New Zealand Biennial Conference on Digital Image & Vision Computing: Techniques and Applications, Auckland, New Zealand, 195-200.

14. Cook, R. L., 1984, “Shade Trees”, Proceedings of the 11th annual conference on Computer graphics and interactive techniques, 223-231.

15. Perlin, K., 1985, “An Image Synthesizer”, ACM Special Interest Group on Computer Graphics and Interactive Techniques, 287-296.

16. Hanrahan, P. & Lawson, J., 1990, “A Language for Shading and Lighting Calculations”, ACM Special Interest Group on Computer Graphics and Interactive Techniques, 289-298.

17. Lastra, A., Molnar, S., Olano, M. & Wang, Y., 1995, “Real-time programmable shading”, ACM Special Interest Group on Computer Graphics and Interactive Techniques, 59-66.

18. Ikedo, T. & Ma, J., 1997, “An Advanced Graphics Chip with Bump-mapped Phong Shading”, 1997, 156-165.

19. Peercy, M., Airey, J., Cabral, B., 1997, “Efficient Bump Mapping Hardware”, ACM Special Interest Group on Computer Graphics and Interactive Techniques, 1997.

20. Heidrich, W., Seidel, H., 1999, “Realistic, Hardware-accelerated Shading and Lighting”, Proceedings of the 26th annual conference on computer graphics, 171-178.

21. McCool, M., Heidrich, W., 1999, “Texture Shaders”, Proc. of the 1999 Eurographics/SIGGRAPH workshop on graphics hardware, 117-126.

22. Peercy, M., Olano, M., Airey, J., Ungar, P., 2000, “Interactive Multi-Pass Programmable Shading”, 425-432.

23. McCool, M., 2001, “SMASH: A Next Generation API for Programmable Graphics Accelerators”, Technical Report CS-2000-14, Computer Graphics Lab, Department of Computer Science, University of Waterloo.

24. Lindholm, E., Kilgard, M., Moreton, H., 2001, “A User-Programmable Vertex Engine”, Proc. of the 28th annual conference on computer graphics and interactive techniques, 149-157

25. Kautz, J., Heidrich, W., Seidel, H., 2001, “Real-Time Bump Map Synthesis”, Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware, 109-114

26. McCool, M., Qin, Z., Popa, T., 2002, “Shader Metaprogramming”, Proceedings of the conference on Graphics hardware 2002, 57-68

27. McAllister, D., Lastra, A., Heidrich,W., 2001, “Efficient Rendering of Spatial Bi-directional Reflectance Distribution Functions”, ACM Special Interest Group on Computer Graphics and Interactive Techniques, 109-114.

28. Robb, M., 2003, “From MCGA to GeForce FX and Radeon – The history of PC Graphics Hardware”, Research Memo CS-2003/01, Heriot Watt University.

29. Frankot, R., Chellappa, R., 1988, “A Method for Enforcing Integrability in Shape from Shading Algorithms”, IEEE Transactions on Pattern Analysis and Machine Intelligence, 10(4), 439-451.

30. Grőller, E., Rau, R., Straßer, W., 1995, “Modeling and Visualization of Knitwear”, IEEE Transactions on Visualization and Computer Graphics, 1(4).

31. Xu, Y., Chen, Y., Lin, S., Zhong, H., Wu, E., Guo, B., Shum, H., 2001, “Photorealistic Rendering of Knitwear using the Lumislice”, ACM SIGGRAPH, 391-398

32. Chen Y., Lin, S., Zhong, H., Xu, Y., Guo, B., Shum, H., 2003, “Realistic Rendering and Animation of Knitwear”, IEEE Transactions on Visualization and Computer Graphics, 9(1).

33. Yasuda, T., Yokoi, S., Toriwaki, J., Inagaki, K., 1992, “A Shading Model for Cloth Objects”, IEEE Computer Graphics and Applications,12(6).

(a) (b) (c) (d)

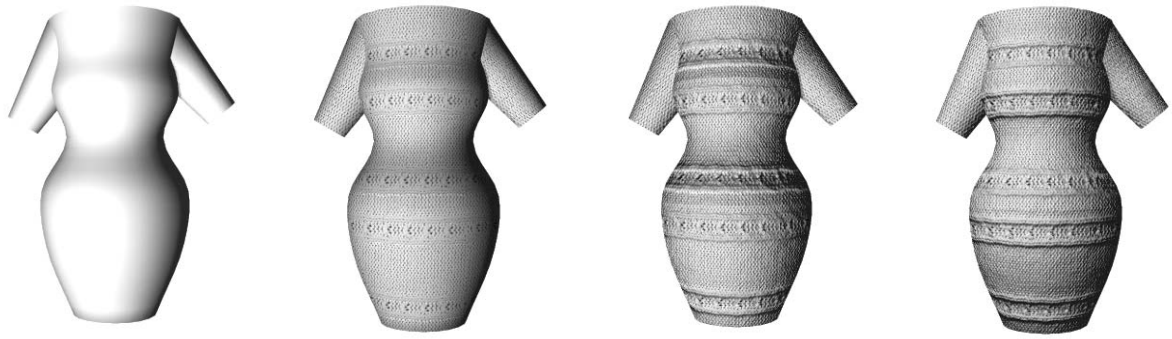

Figure 1. Virtual mannequin (a) geometric model lit from bottom left, (b) texture mapped lit from bottom left, (c) bump mapped lit from bottom left, (d) bump mapped lit from top left. Each rendered in approximately 1 second using Micrografx Simply 3D.

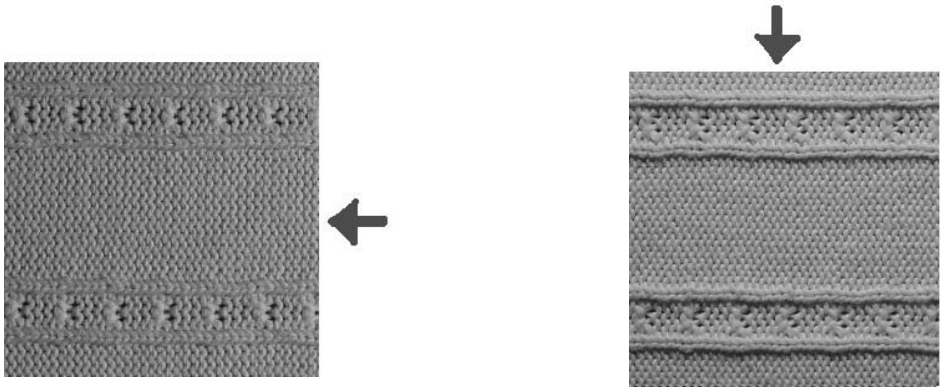

Figure 2. Effect of change in illumination conditions on the appearance of a knitted textile. Arrow indicates direction of illumination.

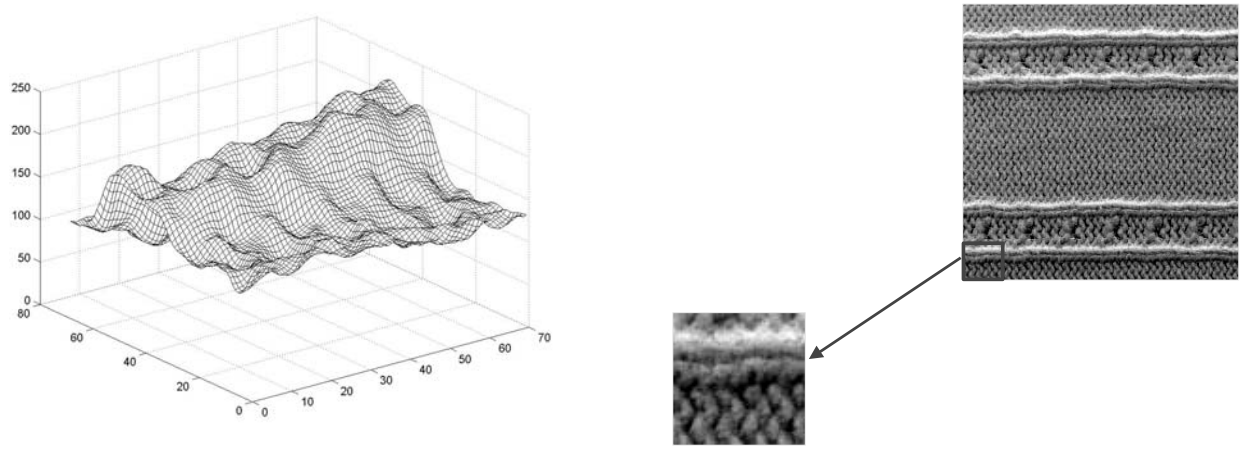

Figure 3. Displacement map obtained by integrating the bump map. Corresponding area of the textile also shown.

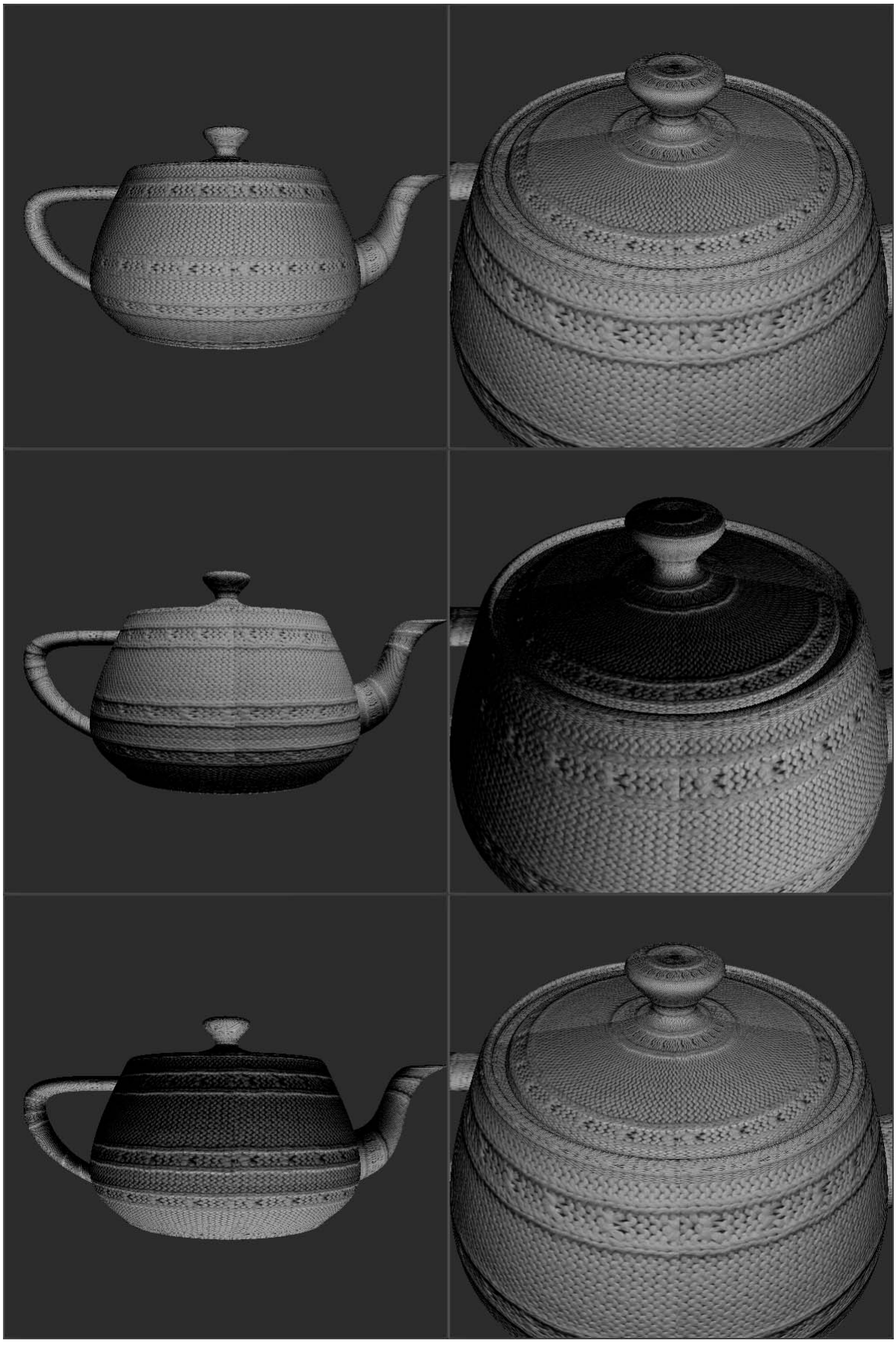

Figure 4. Utah teapot rendered in real-time using a photometrically acquired textile bump-map.