Vision, Video and Graphics (2003)
P. Hall, P. Willis (Editors)

Real-time per-pixel rendering of bump-mapped textures captured using photometric stereo

M. Robb, A. D. Spence, M. J. Chantler and M. Timmins

School of Mathematical & Computer Sciences, Heriot-Watt University,
Riccarton, Edinburgh, EH14 4AS, U.K.

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 visualise selected samples using real-time photorealistic 3D animation. The innovation in this work is 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 albedo map of a surface texture to be captured digitally. When imported into a standard graphics program these images can be used to provide 3D models with a photorealistic appearance. We have developed software that takes advantage of the advanced rendering features of consumer graphics accelerators to produce bump mapped models in real-time. The viewer can manipulate both viewpoint and lighting to gain a deeper perception of the properties of the textile sample.

Categories and Subject Descriptors (according to ACM CSS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism; Colour, shading, shadowing and texture

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.

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 Requisite Information for a Rough Surface

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 surface orientation of an object to the irradiance or 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 – 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 with reasonable accuracy.

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

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.

$$\mathbf{n} = \frac{(-p, -q, 1)^T}{\sqrt{p^2 + q^2 + 1}} \qquad (5)$$

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** as follows.

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

$$\rho = \sqrt{t_1^2 + t_2^2 + t_3^2}$$

For real-time per-pixel rendering, it is these three variables 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 in terms of the partial derivatives. This was generated by a frequency domain integration. The method used in our work is similar to that presented by Frankot & Chellappa [29].

2.1 Optimal capture conditions

McGunnigle [5] proposed using illumination vectors with a spacing of 90° with regard to tilt angle, $\delta\tau$, since this simplifies the scheme given by equation 3 to the extent that it can be solved directly. More recent work of ours [7] has shown that this orientation is not in fact optimal at least for textile textures and a slight improvement in accuracy can be realized by using a spacing of 120°. This was found to the case both empirically and analytically. Our work has therefore employed the more general photometric scheme by Woodham [2].

2.2 Laboratory equipment

A digital camera by Vosskuhler (Model CCD-1300C) was used to capture images of textures under various lighting conditions. A linear light source approximately 1m in distance from the textile sample was used.

3. Real Time Visualisation

3.1 The history of bump-mapping hardware

As mentioned previously, bump-mapping was first introduced by Blinn [1] in 1978, as a means of adding detail by perturbing the normal vector just before lighting calculations However, it was not until the early 1990's that bump-mapping hardware first became available in research laboratories [17,18] then on high-performance workstations [19], and on consumer graphics hardware [28]. Up until that time, research had been directed at extending the flexibility of software rendering systems [14,15,16]. With the availability of low-cost programmable hardware, research has concentrated on implementing both advanced lighting models [20,25,27] and software shading algorithms transparently on graphics hardware [22,23,26]. Research has also led to new extensions being specified [21,24].

Research into the rendering of knit-wear and cloth materials has also been carried out [30,31,32,33]. While 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.

There are several tasks required to visualise the captured images. 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.

3.2 The choice of graphics accelerator

For this project, the NVIDIA GeForce Ti4600 graphics accelerator was selected, as these were readily available from retailers and OEM's. This

was in line with the project objective of implementing real-time rendering using currently available consumer graphics hardware. The specification for this project also required that 3D models could be rendered using at least one local point or directional light source at any time, using both diffuse and specular components. The programmable features of this card make these goals achievable.

3.3 The choice of programming API

In order to gain access to the advanced rendering features of the graphics accelerator, it is necessary to make use of one of the two graphics API's presently available: OpenGL or DirectX. OpenGL is an open standard targeted at the scientific, CAD and entertainment software markets, while DirectX is an API targeted exclusively for the entertainment software market. As the goal of this project is to develop cross-platform software, the OpenGL API was chosen for this application.

3.4 The use of OpenGL extensions

Before we describe the design of the algorithms used for the graphics accelerator, we will describe the capabilities of the card in greater detail. One of the most useful features of the latest OpenGL specification is the ability to replace the transformation, lighting and the rasterisation stages of the graphics pipeline with custom programs. The following extensions were used in the application.

1. Multi-texturing
2. Texture cube-map
3. Vertex programs
4. Register combiners

3.4.1 Multi-texturing

The multi-texturing extensions provide the graphics accelerator the ability to generate the texture colour data from the combination of two or more textures simultaneously. Used alone, this extension only allows the programmer to implement additive or subtractive blending. However, when used with register combiners, it is possible to combine the colour values using more complex mathematical expressions.

3.4.2 Texture cube-mapping

The texture cube-mapping extension gives the graphics accelerator the ability to implement a spherical lookup function based on the value of a single 3D coordinate. Input to such a function can include texture co-ordinates, normal vectors, reflection and refraction vectors. Another use is to implement vector normalisation, where any vector of arbitrary length can be converted into a unit length vector.

3.4.3 Vertex programs

The vertex program extension allows the user to replace the existing OpenGL transformation and lighting stages with a custom transformation stage. This is required to implement the bump-mapping algorithm. The output from the vertex program and multi-texturing extensions is passed to the register combiner stage.

3.4.4 Register combiners

The register combiner extension allows the user to implement per-pixel calculations without having handle the scan-line rasterisation of each triangle. The register combiner stage is comprised of a number of arithmetic multiplexor units, each of which can accept input from a previous stage or from constant registers. The output is generated from various mathematical functions including vector multiplication, addition, linear scaling and power-of-two scaling. The Ti4600 graphics accelerator supports a maximum of eight register combiner units. Applications of register combiners include blending, addition and dot product calculations used for per-pixel specular lighting and bump-mapping.

3.4.5 Other Extensions

There are many other OpenGL extensions that are available, but offered far more functionality than required by this application. A good example is the texture shader extension that allows the user to implement complex calculations by combining the inputs and outputs of texture units together. Examples of such calculations typically include combined diffuse and specular environment cube-mapped texturing and using the output of one texture unit as the input to another. Unfortunately, as our texture acquisition algorithm cannot acquire reflection or specular data, at this time, it is not possible to take advantage of the features that this and other extensions offer.

3.5 Design of the graphics pipeline

Section two described how we obtain the partial derivatives and albedo using photometric stereo. 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 now 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

3.5.1 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 texture 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:

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

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:

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

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 GeForce FX, it will be possible to use the floating-point data directly.

3.5.2 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.

3.5.3 Per-pixel lighting

Per-pixel lighting of the graphics pipeline is implemented using the register combiner unit of the Ti4600 graphics chip. The lighting model used by this implementation incorporates ambient, diffuse and specular components.

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.

3.5.4 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 using the NV_vertex_array_range extension.

3.5.5 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.

4. Conclusions

In this paper, we present a method of photometrically acquiring the image of a rough surface 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. James F. Blinn, Simulation of Wrinkled Surfaces, ACM Special Interest Group on Computer Graphics and Interactive Techniques, 1978, pages 286-292. Also in Tutorial : Computer Graphics : Image Synthesis pp 307 – 313.

2. R. J. Woodham, “Photometric method for determining surface orientation from multiple images” Optical Engineering Jan/Feb 1980 Vol. 19 No.1 pp 139 – 144

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

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

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

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

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

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

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

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

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

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

13. K. Schluns, “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, December 1997, 195-200.

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

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

16. Pat Hanrahan and Jim Lawson, A Language for Shading and Lighting Calculations, ACM Special Interest Group on Computer Graphics and Interactive Techniques, 1990, pages 289-298

17. Anselmo Lastra, Steven Molnar, Marc Olano and Yulan Wang, Real-time programmable shading, ACM Special Interest Group on Computer Graphics and Interactive Techniques, 1995, pages 59-66

18. Tsuneo Ikedo and Jianhua Ma, An Advanced Graphics Chip with Bump-mapped Phong Shading, 1997, pages 156-165

19. Mark Peercy, John Airey, Brian Cabral, Efficient Bump Mapping Hardware, ACM Special Interest Group on Computer Graphics and Interactive Techniques, 1997

20. Wolfgang Heidrich, Hans-Peter Seidel, Realistic, Hardware-accelerated Shading and Lighting, Proceedings of the 26th annual conference on Computer graphics, 1999, pages 171-178

21. Michael D McCool, Wolfgang Heidrich, Texture Shaders, Proceedings of the 1999 Eurographics/SIGGRAPH workshop on Graphics hardware, pages 117-126

22. Mark S. Peercy, Marc Olano, John Airey, P. Jeffrey Ungar, Interactive Multi-Pass Programmable Shading, 2000, pages 425-432

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

24. Erik Lindholm, Mark J. Kilgard, Henry Moreton, A User-Programmable Vertex Engine, Proceedings of the 28th annual conference on Computer graphics and interactive techniques, 2001, pages 149-157

25. Jan Kautz, Wolfgang Heidrich and Hans-Peter Seidel, Real-Time Bump Map Synthesis, Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware, 2001, pages 109-114

26. Michael D. McCool, Zheng Qin and Tiberiu S. Popa, Shader Metaprogramming, Proceedings of the conference on Graphics hardware 2002, pages 57-68

27. David K. McAllister, Anselmo Lastra, and Wolfgang Heidrich, Efficient Rendering of Spatial Bi-directional Reflectance Distribution Functions, ACM Special Interest Group on Computer Graphics and Interactive Techniques, 2001, pages 109-114

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

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

30. Eduard Grőller, René T. Rau, Wolfgang Straßer, “Modeling and Visualization of Knitwear”, IEEE Transactions on Visualization and Computer Graphics, Vol. 1, No. 4, December 1995.

31. Ying-Qing Xu, Yanyun Chen, Stephen Lin, Hua Zhong, Enhua Wu, Baining Guo, Heung-Yeung Shum, “Photorealistic Rendering of Knitwear using the Lumislice”, ACM SIGGRAPH, p391-398, 2001.

32. Yanyun Chen, Stephen Lin, Hua Zhong, Ying-Qing Xu, Baining Guo, and Heung-Yeung Shum, “Realistic Rendering and Animation of Knitwear”, IEEE Transactions on Visualization and Computer Graphics, Vol. 9, No. 1, January-March 2003.

33. Takami Yasuda, Shigeki Yokoi, and Jun-ichiro Toriwaki, Katsuhiko Inagaki, “A Shading Model for Cloth Objects”, Computer Graphics and Applications, IEEE , Volume: 12 Issue: 6 , Nov 1992.

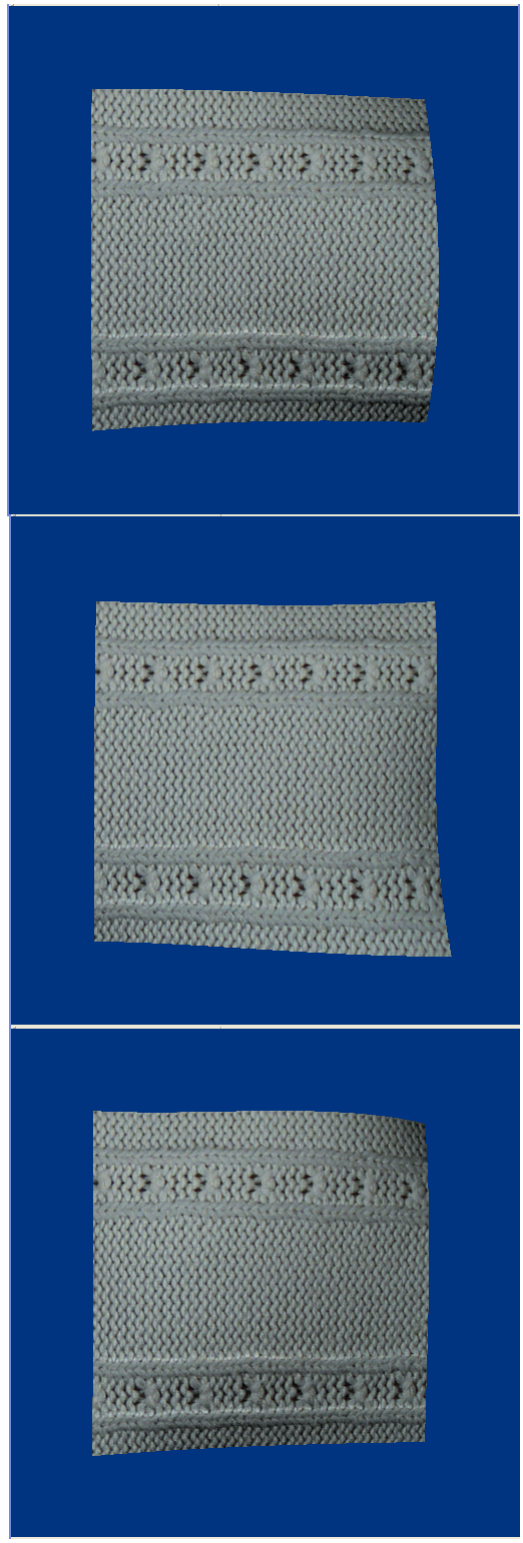

Figure 4: Textured Bezier patch with a knitted textile bump map texture

(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.

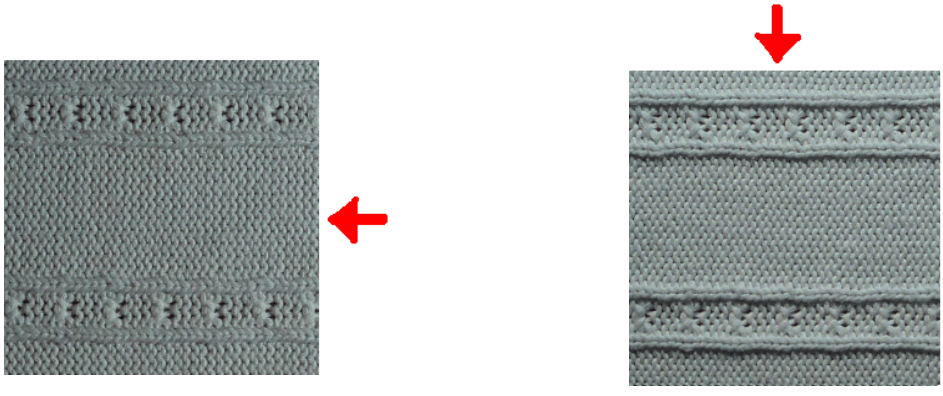

Figure 2 : Effect of change in illumination conditions on the appearance of a knitted textile. Arrow indicates direction of illumination.

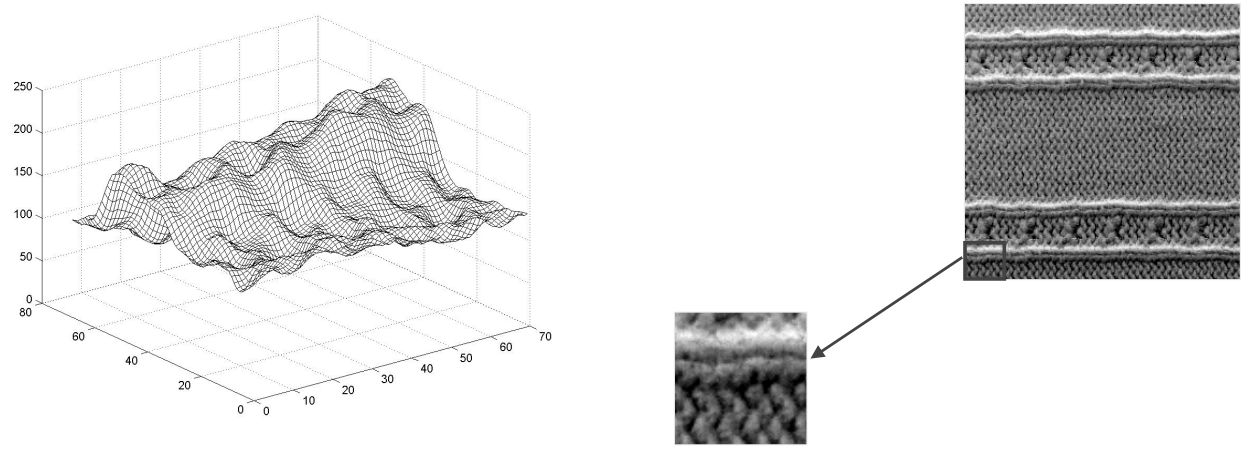

Figure 3 : Displacement map obtained by integrating the bump map. Corresponding area of the textile also shown.