Bump Mapping for Isosurface Volume Rendering

Sergey Belyaev
Peter the Great St. Petersburg Polytechnic University, Polytechnicheskaya, 29, St. Petersburg, 194064, Russia
Email: bel_s_u@mail.ru

Viacheslav Chukanov and Vladislav Shubnikov
Epam Systems, Life Sciences, Chernoi rechki nab, 41, St. Petersburg, 197342, Russia
Email: {kauter1989, vlad.shubnikov}@gmail.com

***Abstract*—In this article a new approach for bump mapping as applied to visualizing medical data using isosurface volume rendering is presented. The method does not require any additional construction and can be used "on the fly" with any method of volumetric rendering. The algorithm is based on classic normal map approach but the normal map and the material's textures are packed as cubic ones and a priori knowledge about the nets of such textures is leveraged to calculate the normal on the fly. The algorithm can be applied to any number of isosurfaces on any gpu that supports cubic textures.**

***Index Terms*—volumetric rendering, medical data visualization, raycasting**

I. Introduction

With the development of medical equipment came the problem of visualizing not only 2D data, but also 3D data. Such images are the result of the reconstruction of data from CT scans, PET scans, and other modern diagnostic methods. The reconstructed images are stored as 3D arrays in the form of a voxel grid. One of the ways to demonstrate data of this kind is to visualize the isosurfaces representing some numeric characteristic of the image's voxels (such as Roentgen density). There are two approaches to visualizing isosurfaces. The first uses a triangular mesh that approximates the required isosurface; a disadvantage of this approach is that it requires the data to be preprocessed in order to construct this approximation. The second approach - isosurface volume rendering - detects this surface "on the fly" by tracing rays passing through the screen's pixels in the given volume. The objective of the work is to improve the quality of the visualization of the isosurface thus obtained.

Since the resolution of the original 3D data is rather low, there is no data on the fine details of various organs, bones, blood vessels, etc. However, this data can be added artificially, so as to make the surface being visualized resemble the real organ, bone, etc. more closely. Common techniques for detailed relief simulation are called bump mapping techniques and some of the underlying ideas can be adapted to a surface being constructed "on the fly".

Manuscript received May 25, 2017; revised November 22, 2017.

II. Reference Overview

In medical visualization, bump mapping is used as follows: first, the isosurface is constructed; then the texture coordinates are calculated; and finally, the texture is applied during visualization. Such an approach requires preprocessing to calculate the geometry of the isosurface [1]. However, for the purposes of medical visualizations the approach most commonly used is isosurface volume rendering, as it can be used to visualize any isosurface without preprocessing [2]-[7].

There are many methods for bump mapping that allow the relief to be modeled by simply changing the normal [8] as well as using more complicated methods that take into account the height of the isosurface and the displacement of the point being displayed relative to the observer [9]-[11]. Most of the bump mapping methods makes use of a normal map, where each point gets assigned to a normal in the tangent space. The tangent space is given by a coordinate system where the z axis is the normal vector N at the point of tangency, while the x and y axes are the vectors T and B, which point along the directions of increase for the texture coordinates u and v respectively and lie in the tangent plane. (See Fig. 1 below)

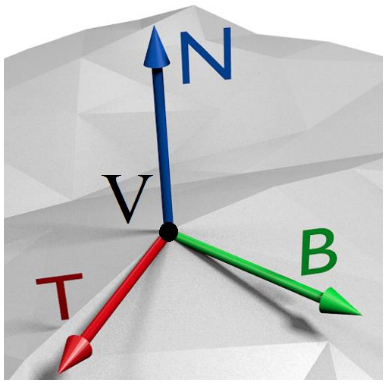


Figure 1. The tangent space. The vectors T and B lie in the tangent plane at V.

To calculate the lighting in the global coordinate system, the texture normal, norm, is normalized to the range [-1, 1] and is converted from the tangent space to the global space with the following formula:

$$n = \begin{pmatrix} tx & bx & nx \\ ty & by & ny \\ tz & bz & nz \end{pmatrix} * \begin{pmatrix} norm_x \\ norm_y \\ norm_z \end{pmatrix} \quad (1)$$

The vector ***n*** thus obtained is used for lighting calculations.

III. Algorithm Description

When isosurface volume rendering is done without preprocessing, there is no additional information on the surface in the voxel grid. The isosurface is visualized, for example, using bounding volume rendering, where the color and transparency of each pixel are determined by moving along the ray with some step length (Fig. 2). In doing this, the voxel grid is placed in the cube with coordinates between 0 and 1, which corresponds to the 3D texture space.

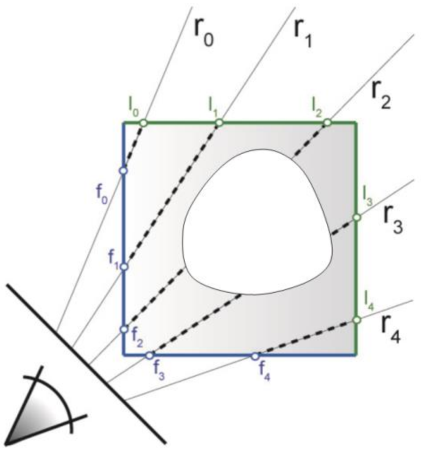


Figure 2. Step-by-step movement along the ray to determine where it intersects the isosurface

In this case, for every point one can find not only its position in the global coordinate system, but also use the values in neighboring voxels to calculate the normal **N** as a gradient.

The algorithm for applying the bump mapping is as follows:

1. Using the 2D normal map, a cubic texture is constructed, where each face is the original 2D image.
2. An analogous texture is constructed for the ***Diffuse*** lighting component.
3. The position of the voxel, **pos**, is renormalized to [-1, 1]:

$$\mathbf{tex} = 2 * \mathbf{pos} - 1 \quad (2)$$

4. The vector **tex** thus obtained is used to pick the color and normal from their respective cubic textures.

The next step is to calculate the vectors of the tangent space. A face of the cube is chosen depending on which component of tex is the greatest in absolute value. In turn, every face of the cube has its own texture space (Fig. 3), which is what determines the basis vectors T and B of the tangent space.

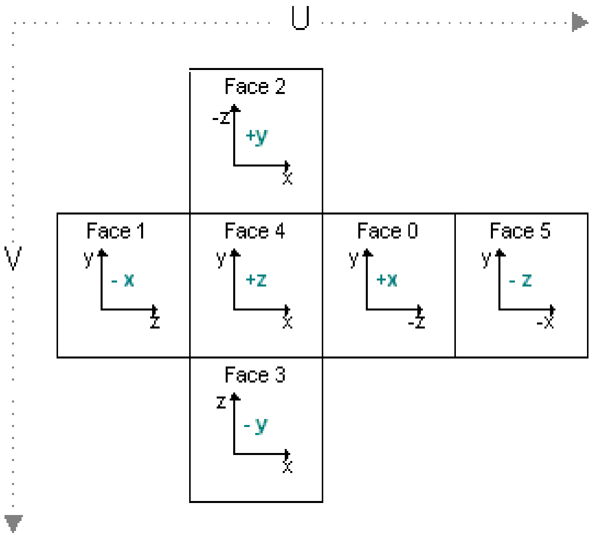


Figure 3. The coordinate systems of every face of the cubic texture (https://msdn.microsoft.com/ru-ru/library/windows/desktop/bb204881(v=vs.85).aspx)

The algorithm to calculate tangent space is based on the isosurface voxel position in 3D texture space, named ***tex*** above. First, the local normal is extracted from the cube map as described below:

$$n = normalize(texture(NormalMap, tex).xyz - vec3(0.5, 0.5, 0.5)) \quad (3)$$

The vector should be unpacked from the range [0..1] to [-1..1]. This technique is very common for storing normal in textures [9] since RGB textures are capable of storing only positive values.

Next is the software handling of cube map texturing process: the same cube map face should be determined as it occurs during texture sampling process. With the face given one can compute tangent and binormal vectors accordingly.

The cube map face to sample a texel from is being selected basing on the maximum absolute value of the input coordinates vector.

Once the basis for the tangent space is determined, the vector from the cubic normal map is converted to the global coordinate system. Since all the vectors are in the same space, a lighting intensity can be computed. The algorithm is not restricted to some specific lighting model, but for simplicity the Phong's model is used:

$$I = Ambient + Diffuse * \max(0, N \bullet L) + \max(0, R \bullet L)^{shininess} \quad (4)$$

In this formula, **Ambient** is the amount of ambient shading, **Diffuse** is the amount of diffused light (this is determined by the texture), **L** is the vector from the point to the light source, **R** is the reflection of **L** about the normal, and **shininess** is the intensity of reflected light.

IV. Results

An algorithm has been implemented for visualizing data using isosurface volume rendering. The computation of lighting has been implemented with a single diffuse color only, also with texturing and normal mapping based on the approach described above. Figs. 4-7 depict the results of the algorithm applied to data obtained from an MRI scan of a human brain. Fig. 4 shows the visualization without bump mapping. Fig. 5 shows the visualization obtained with bump mapping using the presented method. The textures used are shown in Fig. 6. The visualization of the frontal lobe WM is shown in Fig. 7. In this case, only the bump texture is used (the same bump texture as in the previous example).

The algorithm has been tested on PCs with configurations described in the Table I. According to the tests, the texturing and normal computations have almost no effect on the runtime of the whole application: an overhead is approximately 1% of the runtime of the isosurface volume rendering algorithm.

TABLE I. PC Configurarions

	CPU	GPU
Test PC #1	Intel Core I7 2600k 3.4GHz	GeForce 550 Ti 1Gb GDDR5
Test PC #2	Intel Core I7 2600k 3.4GHz	GeForce 470GTX 1Gb GDDR5

V. Conclusion

An algorithm has been presented for relief texturing based on the isosurface volume rendering approach applicable to medical data visualization. The algorithm can run on GPUs that support cubic textures and only increases the workload by 1%. The only extra operations required to perform on GPU are cubic texture sampling and a number of vector operations.

Acknowledgement

We would like to express our sincere appreciation to Applied Math Department of the Peter the Great St. Petersburg Polytechnic University for their ideas, algorithms and infrastructure support. A very special gratitude goes out to all of Epam company managers for their sincere and active support of this research.

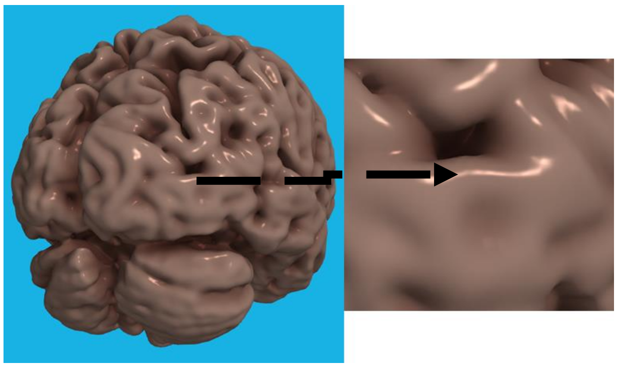


Figure 4. Only the color of the surface is given.

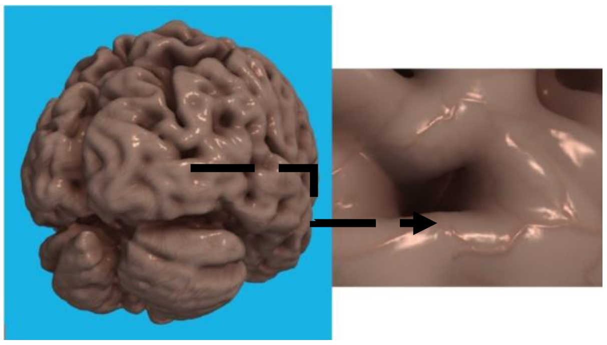


Figure 5. The diffuse and bump textures are used.

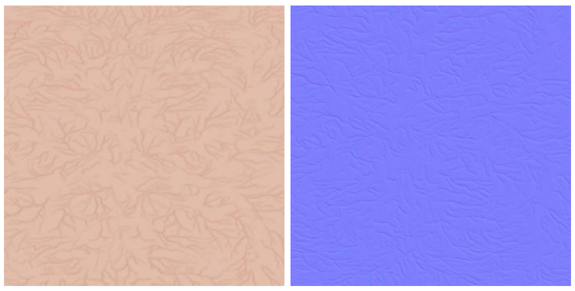


Figure 6. Diffuse (left) and bump (right) textures.

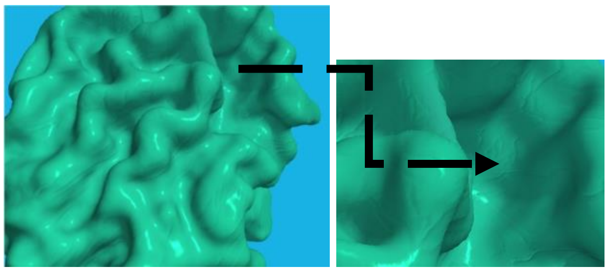


Figure 7. Frontal lobe with bump texturing.

References

[1] M. Wakid, C. Kirmizibayrak, and J. K. Hahn, "Texture mapping volumes using GPU-based polygon-assisted raycasting. (CGAMES)," in *Proc. 16th International Conference on Computer Games, IEEE*, 2011, pp. 162-166.

[2] J. Kruger and R. Westermann, "Acceleration techniques for GPU-based volume rendering," in *Proc. IEEE Visualization*, 2003, pp. 287–292.

[3] V. Bruder, S. Frey, and T. Ertl, "Real-time performance prediction and tuning for interactive volume raycasting," SIGGRAPH ASIA, 2016.

[4] K. Wu, A. Knoll, B. J. Isaac, H. Carr, and V. Pascucci, "Direct multifield volume ray casting of fiber surfaces," *IEEE Trans. of Visualization and Computer Graphics*, pp. 941-949, 2017.

[5] B. Liu, G. J. Clapworthy, and D. F. Isobas, "A binary accelerating structure for fast isosurface rendering," *Computers & Graphics*, vol. 48, pp. 60–70, 2015.

[6] J. Beyer, M. Hadwiger, and H. Pfister, "A survey of GPU-based large-scale volume visualization in EuroVis - STARs, Blinn J. F.: Simulation of wrinkled surfaces," *Proceedings of SIGGRAPH'78*, vol. 12, pp. 286-292, 1978.

[7] B. Preim and C. P. Botha, *Visual Computing for Medicine: Theory, Algorithms, and Applications*, 2nd ed., San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2013.

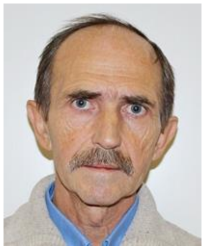
[8] J. F. Blinn, "Simulation of wrinkled surfaces," *Proceedings of SIGGRAPH'78*, vol. 12, pp. 286-292, 1978.

[9] T. Welsh, "Parallax mapping with offset limiting: A perpixel approximation of uneven surfaces," *Infiscape Corporation*, pp. 1-9, 2004.

[10] F. Policarpo, M. Oliveira, and J. Comba, "Real-time relief mapping on arbitrary polygonal surfaces," in *Proc. Symposium on Interactive 3D graphics and Games, ACM*, 2005, pp. 155-162.
[11] F. Policarpo and M. Oliveira, "Relaxed cone stepping for relief mapping," *GPU gems 3*, pp. 409-428, 2007.

Sergey Belyaev was born in Russia, 1954. He received the B.S. and M.S. degrees in Applied Mathematics and Informatics in 1975 and 1977. He's got his Ph.D. degree in Mathematical Modelling, Numerical Methods, and Software Systems from Department of Applied Mathematics, Institute of Applied Mathematics and Mechanics, Peter the Great St. Petersburg Polytechnic University, St. Petersburg, Russia, in 1983.
Since 1983, he has been an Assistant Professor and then an Associate Professor in Department of Applied Mathematics in St. Petersburg Polytechnic University. He has also been working as a Senior Project Manager in EPAM Systems, St. Petersburg, Russia, since 2015. His research interests include computer graphics algorithms.

Viacheslav Chukanov was born in Russia, 1989. He received the B.S. and M.S. degrees in Applied Mathematics and Informatics in 2010 and 2012. He has been participating in many projects on medical image processing, image reconstruction for CT/PET and several projects on computer graphics and visualization. His interests are in the field of computer graphics and image processing.



Vladislav Shubnikov was born in Russia, 1972. He received the M.S. degree in specialization "Applied Mathematics" in 1995. He's got Ph.D. degree by 12 July 2002 in specialization in the Bonch-Bruevich Saint-Petersburg State University of Telecommunications.
Since 1995 he has been an Assistant Professor and later Associate Professor of Applied Mathematics in Peter the Great St. Petersburg Polytechnic University. In parallel he was senior engineer in Driver Inter, Ltd. Since 2015 he also works as a Senior Software Engineer in EPAM Systems, St. Petersburg, Russia. His research interests include image processing, pattern recognition, machine learning, 3d reconstruction and predictive models.