Survey Report on Texture Mapping

Manasi Athale, Sabine Stifter
Research Institute for Symbolic Computation,
Johannes Kepler university, Linz, Austria
E-mail: manasi,stifter@risc.uni-linz.ac.at

Abstract

"A picture is worth a thousand words". This is best portrayed by today's computer graphics systems. Naturally computer graphics researchers have always striven to get most realistic images of real life objects. In 1974, Catmull showed how texture mapping can add reality to computer generated images using texture mapping technique, and since then it is a field of interest for several researchers. In this paper, we give a brief introduction of *texture mapping* concept and its main methods. We describe some of its recently developed methods and also some of the main applications where it is useful. We also discuss popular software used for rendering of images of the real life objects on computer.

1 Introduction

Texture is a physical attribute that characterizes all surfaces. It gives information about the roughness and composition of the surface. It also gives information about size, shape and density of the surface of the object. To generate an image of such object showing all the details is an intriguing problem in computer graphics world as the texture is a difficult visual attribute to synthesize.

In the quest for more realistic imagery, one of the most frequent criticisms of early, synthesized raster images was the extreme smoothness of surfaces, they showed no texture, bumps, scratches, dirt, or fingerprints. Reality demands complexity, or at least the appearance of complexity. Texture mapping is a relatively efficient means to create the appearance of complexity without the tedium of modeling and rendering every three-dimensional detail of a surface [H86].

Texture mapping is a technique, which increases the visual complexity of a scene without increasing its geometric complexity. The idea is that the rendering system maps an image onto simple scene geometry to make objects look much more realistic than the underlying geometry.

The study of texture mapping is valuable because its methods are applicable throughout the computer graphics and image processing. Geometric mappings are relevant to the modeling of parametric surfaces in CAD and to general two-dimensional image distortions for image restoration and artistic uses. The study of texture filtering leads into the development of space variant filters, which are useful for image processing, artistic effects, depth-of-field simulation, and motion blur [H86].

In 1974, Catmull introduced texture mapping and since then a great deal of work has been devoted towards the improvement of the quality of the generated images and the reduction of the computational cost. In general, the problem of texture mapping can be stated as follows: given an arbitrarily curved surface and a texture domain, a (at least locally inevitable) mapping between these two domains. The texture domain is one-dimensional, two-dimensional or three-dimensional and may include color, roughness or transparency entries [AA00].

When mapping an image onto an object, the color of the object at each pixel is modified by a corresponding color from the image. In general, obtaining this color from the image conceptually requires several steps [H89]. The image is normally stored as a sampled array, so a continuous image must first be reconstructed from the samples. Next, the image must first be warped to match any distortion in the projected object being displayed. The warped image is then filtered to remove high frequency components that would lead to aliasing in the final step: sampling again to obtain the desired color to apply to the pixel being textured [HS93].

There are several methods in use for texture mapping and several different approaches have been suggested to achieve high quality in image synthesis. In the next section we describe them as they got developed in years.

2 Texture Mapping

Though one strives to add reality to images produced on the screen, it is not that simple. We would like to quote from [F], "Visual forms - lines, colors, proportions, etc. - are just as capable of articulation, that is, of complex combinations, as words but the laws that govern this sort of articulation are altogether different from the laws of syntax that govern language.... They do not present their constituents successively, but simultaneously, so the relations determining a visual structure are grasped in one act of vision." This is the motivation and also inspiration to all researchers in computer graphics field for generating realistic images.

Texture mapping is an important technique for improving the reality of objects rendered on computer. Using texture mapping techniques, the appearance of detail can be added to an object without increasing the amount of geometry needed to model the object. We give here a brief overview of texture mapping basics and some of the methods.

2.1 What is texture mapping?

Every object in nature has a specific size, shape, color, density and texture. To generate its image on the computer, one has to consider each of the above characteristics. The characteristics like size and shape of the object are easy to manipulate on the screen than its texture. For example, a stone or a piece of wood has different consistency at each point. It is not smooth at every point, or have definite structure.

A *texture* means a detailed pattern that is repeated many times to tile the plane, or more generally, a multi-dimensional image that is mapped to a multi-dimensional space. *Texture mapping* means the mapping of a function onto a surface in three-dimensional space [H86]. The domain of the function can be one, two, or three-dimensional space. This can be represented by an array or a mathematical function. One-dimensional texture can simulate rock layer two-dimensional texture can represent waves, and three-dimensional texture can represent wood.

In other words, *texture mapping* is the process of shaping a texture image and applying it to the surface of a geometric primitive, [KL00]. We give here an example of three-dimensional logo. We can explain this better with an example. A plain logo of *RISC* and map a texture on it to give it a three-dimensional look. Following figure is created in *CorelDraw 9*, and then a texture is mapped onto it using *Ulead COOL 3D 3.0, trial version.*

(a) Plain Logo

(b) Logo after texture mapping

Figure 1: Application of texture mapping on a hypothetical logo

Using the similar technique, we can create a three-dimensional logo. The figure:1(b) gives an impression of how a three-dimensional logo looks like which is created using texture mapping on plain logo in figure:1(a).

2.2 Texture Mapping Methods

The basic idea of texture mapping methods is that the rendering system maps an image onto simple scene geometry to make objects look much more realistic than the underlying geometry. We describe here some of the basic methods for texture mapping [TM].

(Flat) Texture Mapping: Mapping the texture onto the scene objects is done at the end of the rendering process. Once it is determined which pixels are covered by the object on the screen each square S representing one of those pixels is back-projected onto the three-dimensional object. Then, in a second transformation, the area on the texture image A corresponding to S is found. The color of S is assigned to be some average of all pixels in A. In order to find this average a number of re-sampling filters can be used.

Bump Mapping: Flat textures just modify the surface colors, but not the light reflection patterns. Bump maps have for every pixel also a vector or an index to a vector stored, which is used to perturb the surface normal at the corresponding location of the textured object. In the shading stage of the rendering pipeline Phong shading has to be used. Thus for each screen pixel the illumination equation is evaluated, each time with a new normal slightly modified due to the bump map. The effect is that rough surfaces look a lot more realistic most of the time. Along the edges of an object it is still visible that the underlying geometry is actually flat.

Reflection Mapping: This technique requires two passes through the rendering pipeline. First the viewpoint is transformed to the virtual projection center inside the object O to be textured. Now the scene of all other objects is rendered and the resulting image is stored as texture image of O. In the second pass the scene is rendered with the original viewpoint and the Surface of O will now show the reflection of its environment.

Displacement Mapping: This method is very similar to the bump mapping. Just in this case the vectors represent perturbations of the normal vectors and the actual surface. So now even the edges of curved objects can appear a lot more complex than the geometric representation. The price for this added realism is the need to apply the texture map in object space before visible surface determination.

MIP mapping (Multo In Parvo): This method stores texture maps of multiple resolutions efficiently. This Store RGB components in three quarters of the MIP map and four times smaller averaged versions of the RGB components in the remaining quarter. It is then recursed until the remaining quarter is only one pixel. Computing the color for a screen pixel of a texture mapped object is done by:

- finding the area on the texture map affecting the pixel,
- selecting the two resolutions of the MIP map in which four pixel squares cover just a little more or less than the corresponding screen pixel,
- trilinearly interpolate the MIP map pixel values to get the screen pixel color.

The researchers have developed various methods for texture mapping since the introduction of the texture mapping methods in 1974 by Catmull. This is done using sophisticated hardware as well using software. For hardware, consider [CF00, GMW01, GMW00].We here concentrate only on texture mapping methods using software.

3 Applications of Texture Mapping Methods

Texture Mapping methods are used in variety of applications to improve the quality of rendered images. We here give some of the widely known applications in brief.

In basic texture mapping, an image is applied to a polygon (or some other surface facet) by assigning texture coordinates to the polygon's vertices. These coordinates index a texture image, and are interpolated across the polygon to determine, at each of the polygon's pixels, a texture image value. The result is that some portion of the texture image is mapped onto the polygon when the polygon is viewed on the screen. Typical two-dimensional images in this application are images of bricks or a road surface (in this case the texture image is often repeated across a polygon); a three-dimensional image might represent a block of marble from which objects could be "sculpted".

Projective Texture Mapping: A generalization of this technique projects a texture onto surfaces as if the texture were a projected slide or movie. In this case, the texture coordinates at a vertex are computed as the result of the projection rather than being assigned fixed values. This technique may be used to simulate spotlights as well as the re-projection of a photograph of an object back onto that object's geometry. Projective textures are also useful for simulating shadows. In this case, an image is constructed that represents distances from a light source to surface points nearest the light source. This image can be computed by performing Z-buffering [M] from the lights point of view and then obtaining the resulting Z-buffer. When the scene is viewed from the eye-point, the distance from the light source to each point on a surface is computed and compared to the corresponding value stored in the texture image. If the values are (nearly) equal, then the point is not in shadow otherwise it is in shadow. This technique should not use MIP-mapping, because filtering must be applied after the shadow comparison is performed.

Image Warping: Image warping [H89] may be implemented with texture mapping by defining a correspondence between a uniform polygonal mesh (representing the original image) and a warped mesh (representing the warped image). The warp may be affine (to generate rotations, translations, shearing, and zooms). The points of the warped mesh are assigned to the corresponding texture coordinates of the uniform mesh, and the mesh is texture mapped with the original image. This technique allows interactive image warping which could be easily controlled. The technique can also be used for panning across a large texture image by using a mesh that indexes only a portion of the entire image.

Transparency Mapping: Texture mapping may be used to lay transparent or semi-transparent objects over a scene by representing transparency values in the texture image as well as color values. This technique is useful for simulating clouds and trees for example, by drawing appropriately textured polygons over a background. The effect is that the background shows through around the edges of the clouds or branches of the trees. Texture map filtering applied to the transparency and color values automatically leads to soft boundaries between the clouds or trees and the background.

Surface Trimming: This technique is similar to transparency Mapping. This may be used to cut holes out of polygons or perform domain space trimming on curved surfaces. An image of the domain space trimmed regions is generated. As the surface is rendered, its domain space coordinates are used to reference this image. The value stored in the image determines whether the corresponding point on the surface is trimmed or not.

4 Different approaches of texture mapping along the time-line

The research in texture mapping field is growing very fast. Many researchers have developed different methods and different approaches have been suggested so far. Many methods are developed particularly for some specific application. We describe here some of the recently suggested methods for texture mapping using software . Transparent, smoothly curved surfaces can be rendered by using stroke texture method [FIP97]. Transparency can be a useful device for showing multiple overlapping surfaces in a single image. The main problem is to render the transparent surfaces in such a way that their three-dimensional shape can be readily understood and their depth distance from underlying structures can be clearly perceived. This approach shows that the use of sparsely-distributed discrete, opaque texture can be used for more explicitly indicating the relative depth of a transparent surface and for communicating the essential features of its three-dimensional shape. This is done by using stroke texture that is locally oriented in the direction of greatest normal curvature and in which individual strokes are of a length proportional to the magnitude of the curvature in the direction they indicate. This is particularly helpful in radiation therapy, one of the three principal treatments for the control and/or cure of cancer. This method can be used effectively in the visualization of layered surfaces in radiation therapy treatment planning data.

The calculation of detailed shadows is one of the most difficult challenges in computer graphics, especially in the case of extended (linear or area) light sources. In [SS98] a new tool is introduced for the calculation of shadows cast by extended light sources. Exact shadows are computed in some constrained configurations by using a convolution technique, yielding a fast and accurate solution. Approximate shadows can be computed for general configurations by applying it to a representative ideal configuration. For fast shadow calculation in arbitrary configurations a hierarchical, error-driven algorithm is derived using a hierarchy of object clusters. The convolution is performed on images rendered in an off-screen buffer and a shadow map is produced, which is used as a texture to modulate the unobstructed illumination. Light sources can have any three-dimensional shape as well as arbitrary emission characteristics, while shadow maps can be applied to groups of objects at once. The method can be employed in a hierarchical radiosity system, or directly as a shadowing technique.

When rendering a point on a textured surface, the parameters of the surface are normally used to index the texture table in order to determine the texture of the point being rendered. In bump mapping, the methods derived for ordinary texture mapping can not be trivially applied, as it is subject to more geometric constraints and complexity. In [Z98] a new approach is introduced for least texture distortion in bump mapping for a free form surface patch. This is done by establishing a mapping between the parameter space and texture space using techniques of surface development, and texturing by re-parameterization of the bump mapping function using the mapping established at the first step. This can then reduce the texture distortion of bump mapping as much as that of ordinary texture mapping achieved by the most effective least distorted texture mapping methods.

In projective texture mapping, a texture is projected onto the geometric primitive as if it is a projected slide or a movie. In this case, the texture coordinates at a vertex are computed as the result of the projection rather than being assigned fixed values. This technique is first introduced by Mark Segal. Even though it is a powerful tool for image-based rendering, it needs visibility information in order to project textures onto the geometric primitive.

A novel visibility algorithm is suggested in [Y99]. It is a hybrid algorithm, operating in both image-space and object-space. It has object-space precision, but part of it operates image-space in order to get better performance. It generates less polygons than traditional object-space algorithms to help accelerate the speed of texture mapping. The techniques used to achieve these are conservative image-space occlusion testing and object-space shallow clipping.

Texture mapping techniques are not only used in scientific applications, but also to develop art form. An interesting method of image processing to kaleidoscopic image is discussed in [FM99]. The Iamascope is an interactive, electronic Kaleidoscope. This combines video, graphics, vision, and audio technology for performers to create striking imagery and sound with this aesthetically uplifting interactive device. In this installation, user takes the place of a colorful piece of floating glass inside the Kaleidoscope, and simultaneously views a Kaleidoscopic image of themselves as a huge screen in real time. The image processing uses simple intensity differences over time which are calculated in real time. The responsive nature of the whole system allows users to have an intimate, engaging, satisfying multimedia experience. Texture mapping is used to create a flexible, interactive Kaleidoscope. The techniques used to create such Iamascope are well explained in [FM99]. It is claimed that the texture mapping methods used for this application are real time efficient. We would like to explore this technique in our application.

A two stage method for applying texture from a two-dimensional domain to three-dimensional domain is given in [AA00]. It consists of pre-processing and mapping stages. In pre-processing, proper planar development of the three-dimensional surface is generated. In the next stage, that is, in mapping stage, texture is mapped to the given surface. It is proposed that surface flattening technique is well suited compared with other methods which are in use, as in the surface flattening technique the local mapping accuracy can be controlled. This needs a new mapping method and a set of new indices for evaluating the mapping accuracy. A numerical method of singular-value decomposition is used to derive accuracy indices. A space-variant filter is developed to reduce aliasing. This can be incorporated within the two stages within the two stages of proposed method. This could be useful to improve the speed of texture mapping for static pictures. This is interesting feature for us.

A good compromise technique between the mapping quality and the mapping control is suggested in [BS00]. This method is suggested for polygonal models. In this method, a B-spline patch is introduced to control the parameterization of the polygonal model. For each region to be mapped, the algorithms first constructs a B-spline patch with similar shape to surround the model. The mapped region is then projected onto the constructed B-spline patch to achieve a parameterization. By interactively controlling the B-spline patch, the user can conveniently decorate the surface of the model to meet its requirements. t is also suggested that an efficient method could be developed to locally update the texture coordinates using hierarchical B-spline editing techniques which would be helpful in our application.

Though algorithms exist for synthesizing a wide variety of textures over rectangular domains, it is still difficult to synthesize general textures over arbitrary manifold surfaces. In [LW01], a robust solution to this problem for surfaces defined by dense polygon meshes is given. For each mesh vertex, a local parameterization surrounding the vertex is established, and this is used to create a small rectangular neighborhood with the vertex at its center, and a sample texture is searched for similar neighborhoods. The given algorithm requires only a sample texture as input and a target model. It does not require specification of a global tangent vector field. It computes either randomly or via a relaxation process. Despite this, the synthesized texture contains no discontinuities, exhibits low distortion, and is perceived to be similar to the sample texture.

A new texture filtering technique called Feline (for Fast Elliptical Lines) is given in [FJMP99]. Feline uses an underlying space-invariant (isotropic) filter with MIP-mapped data, and so can be built on top of an existing trilinear filtering engine. To texture a pixel, it uses this space-invariant filter at several points along a line in texture space, and combines the results. By increasing some complexity in implementation, Feline more accurately matches the desired projection of the pixel filter in texture space, resulting in images with fewer aliasing artifacts. Feline's visual quality compares well against Elliptical Weighted Average, the best efficient software anisotropic texture filtering algorithm known to date, but Feline requires much less setup computation and far lesser cycles for texel fetches. As it uses standard mip-maps, Feline requires minimal extensions to standard three-dimensional interfaces like OpenGL.

In [GHSS01], a method presented, for a given arbitrary mesh, to construct a progressive mesh (PM) such that all meshes in the its sequence share a common texture parametrization . The method first minimizes texture stretch (small texture distances mapped onto large surface distances) to balance sampling rates over all locations and directions on the surface. It also minimizes texture deviation to obtain accurate textured mesh approximations. The method begins by partitioning the mesh into charts using planarity and compactness heuristics. It creates a stretch-minimizing parametrization within each chart, and resizes the charts based on the resulting stretch. It then simplifies the mesh while respecting the chart boundaries. The parametrization is re-optimized to reduce both stretch and deviation over the whole progressive mesh sequence and the charts are packed into a texture atlas.

Texture mapping is achieved by using some specifically developed hardware tools. In [CF00], a hardware accelerated approach is presented for the voxelization of a wide range of three-dimensional objects, including curves or surfaces, solids, and volumetric CSG models. It can model and manipulate three-dimensional scenes in their own representations, and can generate the volume representations of regions of interest on-the-fly for volumetric operations. The algorithms take advantages of graphics hardware features in modern graphics workstations, such as texture mapping and frame buffer blending functions, to generate slices of the volume representations for various three-dimensional models. As a result, interactive object manipulations and boolean operations can be performed in an intermixed environment of geometric and volumetric objects under a unified volume graphics framework.

5 Note on referred material

We consulted many recent master's and doctoral theses from the universities around the world [H89, L91, D97, B00, O00, S01]. We also studied many recent journal publications and conference papers. To keep the bibliography concise and selective, we only mention most relevant papers. Most of the papers are taken from Computer Graphics Journal and SIGGRAPH proceedings from electronically available sources. The lecture notes [F, M], and books [DF84, DFFHP94] helped a lot to get acquainted with the topic and also pointed to the different resources on the topic. The mathematical concepts, which are used in the texture mapping methods, are well explained in [S]. Few theses were particularly helpful for clear understanding of the basics of texture mapping, for example, [H89].

References

[AA00] P. N. Azariadis, N. A. .Aspragathos. **On using planar developments to perform texture mapping on arbitrarily curved surfaces.,** *Computer and Graphics, 24 (2000), pp. 539–554.*

[BS00] H. Sun, H. Bao. **Interactive texture mapping for polygonal models,** *Computational Geometry 15 (2000) pp. 41–49.*

[B00] J. Bolliger. **A framework for network-aware applications,** *ETH Zürich, Institute of Computer Systems, April 2000.*

[CDV98] L. Darsa, B. Costa, A. Varshney. **Walkthroughs of Complex Environments Using Image-Based Simplification,** *Computer and Graphics, 22 (1998), pp. 55–69.*

[CF00] S. Fang, H. Chen. **Hardware Accelerated Voxelization,** *Computer and Graphics, 24 (2000), pp. 433–442.*

[D97] J. S. De Bonet. **Novel Ststical Multiresolution Techniques for Image Synthesis, Discrimination and Recognition,** *Master Thesis, Department of Electrical Engineering and Computer Science, May 1997.*

[DF84] J. D. Foley, A. van Dem. **Fundamentals of Interactive Computer Graphics,** *Addison Wesley, 1984.*

[DFFHP94] J. D. Foley, A. van Dem, S. K. Feiner, J. F. Hughes, R. L. Phillips. **Introduction to Computer Graphics,** *Addison Wesley, 1994.*

[DG01] J. M. Dischler, D. Ghazanfarpour. **A Survey of 3D Texturing** *Computer and Graphics, 25 (2001), pp. 135–151.*

[F] M. Firebaugh. **Computer Graphics,** *Lecture Notes, CSci 320.* `http://www.uwp.edu/academic/computer.science/Faculty/firebaugh.www/LectIndex320.html`

[FIP97] H. Fuchs, V. Interrante, S. M. Pizer. **Conveying the 3D Shape of Smoothly Curving Transparent Surfaces via Texture,** *IEEE Computer Graphics and ApplicationsI, June 1997.*

[FJMP99] K. I. Farkas, N. P. Jouppi, J. McCormack, R. Perry. **Simple and Table Feline: Fast Elliptical Lines for Anisotropic Texture Mapping,** *WRL Research Report 99/1.*

[FM99] S. Fels, K. Mase. **Iamascope: a graphical musical instrument,** *Computer and Graphics, 23 (1999), pp. 277–286.*

[GHSS01] S. J. Gortler, H. Hoppe, P. V. Sander, J. Snyder. **Texture Mapping Progressive Meshes.**

[GMW00] D. Gelb, T. Malzbender, K. Wu. **Polynomial Texture Maps,** *Hewlett Packard Laboratories, April 2000.*

[GMW01] D. Gelb, T. Malzbender, K. Wu. **Light-Dependent Texture Mapping,** *Hewlett Packard Laboratories, April 2001.*

[H86] P. Heckbert. **Survey of Texture Mapping,** *IEEE Computer Graphics and Applications, Nov 19986, pp. 56–67.*

[H89] P. Heckbert. **Fundamentals of texture mapping and image warping,** *Master Thesis, Carnegie Mellon University, 1989.*

[HS93] P. Haeberli, M. Segal. **Texture Mapping as a Fundamental Drawing Primitive, GRAPHICA***Obscura, June 1993.*

[KL00] L. Knepper, T. Lanzoni **Texture Mapping,** *White Paper, February 2000,* `http://www.dell.com/us/en/biz/topics/vectors_2000-texturemapping.htm`

[L91] R. C. Lansdale. **Texture Mapping and Resampling for Computer Graphics,** *Master Thesis, Department of Electrical Engineering, University of Toronto, Toronto, Canada.*

[LW01] Li-Yi Wei, Marc Levoy. **Texture Synthesis over Arbitrary Manifold Surfaces,** *Stanford University.*

[M] H. Mayr. **Virtual Environments: Design, Modeling, Visualization and Simulation,** *Lecture Notes, 2000-01.*

[O00] E. Oswald. **Generic 2D Graphics Interface with Object Framework and Applications,** *ETH Zürich, Institute of Computer Systems, December 2000.*

[S] S. Stifter. **Geometric Foundations for Symbolic Computation,** *Lecture Notes, 2000-01.*

[SS98] F. X. Sillion, C. Soler. **Fast Calculation of Soft Shadow Textures Using Convolution,** *SIGGRAPH 1998.*

[S01] **Multiresolution Representation and Compression of Surfaces and Volumes,** *ETH Zürich, Institute of Scientific Computing, February 2001.*

[Y99] Yizhou Yu. **Efficient visibility processing for projective texture mapping,** *Computer and Graphics, 23 (1999), pp. 245–253.*

[Z98] J. J. Zhang. **Least Distorted Bump Mapping Onto Surface Patches ,** *Computer and Graphics, 22 (1998), pp. 233–242.*

[TM] `http://www.ecsl.cs.sunysb.edu/cse659/tm.html`