Vision, Video and Graphics (2005)
E. Trucco, M. Chantler (Editors)

Texture Mapping Volume Objects

P. Shen[†] and P. Willis[‡]

Department of Computer Science, University of Bath, BATH BA2 7AY, UK

Abstract

We present a combination of image-based texture mapping and projective space (pseudo-solid) texture. This image-based texture mapping is useful for objects defined from volume datasets. The paper makes three main contributions. First, it introduces the combination of the image-based two-part texture mapping and projective space texture mapping for volume objects. Second, it presents a multi-resolution technique to overcome problems with projecting at glancing angles and to eliminate artifacts due to the resolution limitations. Third, it presents the pixel-level data-dependent interpolation technique in projective image warping. The proposed approach leads to superior quality of texture and thus provides an optional solution for texturing volume objects.The results show the effectiveness and quality of rendered images.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modelling I.3.7 [Computer Graphics]: Three Dimensional Graphics and Realism

1. Introduction

Applying textures to solids has traditionally been performed in one of two distinct ways. One way is to construct a parametric map of the surface, and use the resulting *uv*-parameterization to modify the appearance. An alternative approach is to define a function, which generates a texture value everywhere in space.

Modelling and adjusting the *uv*-representation all over the surface of a model is quite difficult and thus tedious for artists or other users but the use of an image for the texture is attractively simple. In contrast, basic solid textures [Pea85] are easily defined in space but must be expressible in mathematical form. Hypertextures could texture complex volume objects [PH89, SJ01], animators might still need to construct extra density modulation functions for realistic texture mappings.

The work presented in this paper is motivated by projecting image-based 2D texture maps onto volume objects. We approach this in the context of the works on discretely sampled object representations (DSORs) [CIJ*05]; that is, volume datasets and similar forms.

[†] P.Shen@bath.ac.uk
[‡] P.Willis@bath.ac.uk

Solid textures are of course particularly suited to volume objects since they do not need *uv*-parameterization and the concept of surface is less important for such data. They also define a texture representation everywhere in space. However, the limits on modelling the mathematical expression of solid texture are not helpful to us.

We believe that 2D texture maps still play an important role in graphical modelling and thus have value in the specification of complex volumetric scenes. In fact, they can be used together with solid textures, as Winter showed in his PhD thesis [Win02].

Therefore, in this paper we address again the issue of this intermixing of the two approaches. We use plenoptic indexing functions to substitute the mathematical/procedure models for projective space (pseudo-solid) texturing. We show how our approach avoids degradation of rendered image quality, while interpolating texture to resolutions which exceed that of the texture map.

2. Related Works

A texture mapping process, which does not require uv-parameterization, was presented by Bier and Sloan [ES86] in 1986. As the eponym "two-part texture mapping" suggests, the texture is first mapped onto a bounding intermediate surface and is then projected onto the object as a second step.

Intermediate surfaces tend to be simple geometrical primitives such as a sphere, cylinder or cube.

Solid texture was presented by Perlin [Per85] and Peachey [Pea85] independently in 1985. It is designed for overcoming the limitations of parametric texture-mapping techniques for complex surfaces, which may not have parametric forms. Only a location in space of a surface point needs to be determined for each screen pixel onto which the surface projects. A texture value is established by evaluating a procedural spatial texture function. This eliminates surface color discontinuities occurring as a result of poorly-drawn textures or poorly-defined surface parameters. Procedure solid textures by definition have infinite detail and are, therefore, suitable for close up viewing where texture errors, seams or aliasing would otherwise be noticeable.

However image-based texture maps have their advantages too. They do not require a procedural definition but only expect the user to provide a pixel map containing the texture. This supports a very general class of textures but always at a fixed resolution. There are techniques for enhancing the resolution of an image, so there is promise in using a texture map, for its generality, and interpolating higher-resolution texture on demand. Candidate techniques include bilinear, bicubic or cubic B-spline interpolation; and feature-based methods such as edge directed interpolation (EDI) [AW96] and the new edge directed interpolation (NEDI) [LO00]. These could be used to regenerate the texture image with high resolution. Su and Willis [SW04] [SW03] described a fast method which takes edge information into account, in order to retain visual sharpness when the image resolution is increased. In their method, high resolution images are interpolated from the pixel level data-dependent triangulation of lower resolution images. The triangulation is arranged to follow high-contrast image features. The resulting interpolation is in continuous space and thus is suitable for arbitrary resolution enhancement of a still image, with possibly varying resolution across the image. As we will describe, these features are very useful for volume texture mapping.

3. Volume Texturing

The purpose of our texture mapping is to give the surface of a volume object a realistic appearance. Since we avoid *uv*-parameterization for such color-position indexing, two-part texture mapping technique is adopted [ES86].

In two-part texture mapping (projection mapping) only the locations of 3D points in Euclidean space within the bounds of the volume object are needed. Thus it is possible to determine these locations during volume rendering (Direct Volume Rendering or Direct Surface Rendering [Jon99]) and assign the projective intensity of texture to the voxels of the volume object.

Figure 1 gives the pipeline of our methods. In step (2), given the volume object (1) in 3D space, using a volume rendering (Direct Volume Rendering or Direct Surface Rendering [Jon99]) technique, we project this volume object onto a plenoptic surface (cylinder, sphere or cube). This rendered plenoptic surface is then converted into a 2D-rectangle image, which we call the *intermediate template*.

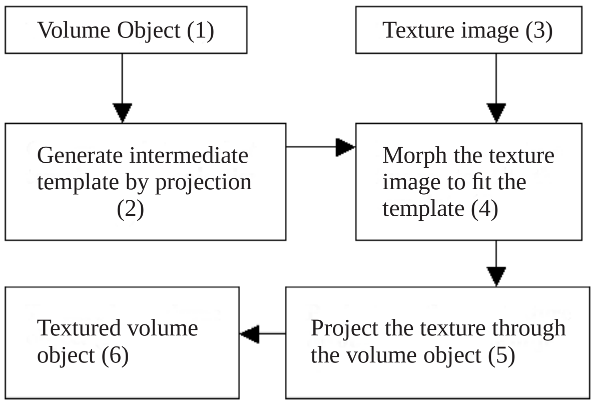

Volume Object (1)
Texture image (3)
Generate intermediate template by projection (2)
Morph the texture image to fit the template (4)
Textured volume object (6)
Project the texture through the volume object (5)

Figure 1: *The pipeline of the two-part texture mapping for volume object.*

A continuous position (normalized) in the intermediate template is defined as (u,v) . The discrete position of a pixel in the intermediate template is defined as (x,y) . The integral coordinate pair (x,y) is indexed by u, v , where $u=x/(x_d-1)$, $v=y/(y_d-1)$, x_d and y_d are the dimensions (the width and the height) of the intermediate template, $0 \leq x \leq x_d, 0 \leq y \leq y_d$.

In step (3), a 2D plenoptic projected image could be acquired using software tools (for instance, the Cyberware Portrait Scanner), or an image could be warped to construct a 2D plenoptic projected image. In step (4), the 2D plenoptic projected image generated in step (3) is morphed according to the intermediate template generated in step (2). We call this image the *morphed texture image*. In step (5), the morphed texture image generated in step (4) is inversely projected onto the volume object. Step (6) is the generated textured volume object.

The positions of the high-contrast texture features in the intermediate template are manually picked up and are employed as control features, for constructing the warping and morphing operations in steps (3) and (4).

As shown in Figure 1, there are six elements in our image-based texture mapping pipelines. Examples of the generated images of these six elements are given in Figure 2. Figure 2(a) is the original volume object. Figure 2(b) is the intermediate template. Figure 2(c) is the plenoptic projected texture image. Figure 2(d) is the morphed texture image. Figure 2(e) is the textured volume object.

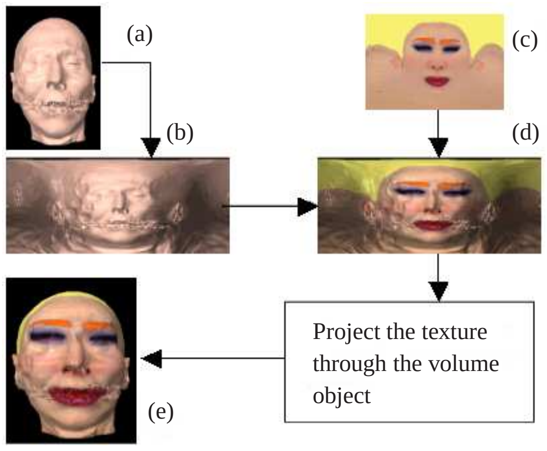


Figure 2: *Image-based texture mapping for volume object: (a) The original volume object. (b) The intermediate template. (c) The plenoptic projected texture image. (d) The morphed texture image. (e) The textured volume object.*

3.1. Intermediate Template

The intermediate template generated in step (2) sets up the connections between the voxels/positions within the volume object and the positions in the 2D plenoptic projected image generated in step (3). Therefore, it is not only an image generated on a geometrical primitive (cylindrical, spherical or cubic surface), as is usual for conventional two-part texturing mapping, but also a rendered image whose rendering-engine is based on cylindrical mapping, spherical mapping or cubic mapping. In other words, the origins of the tracing rays in this rendering engine locate on the cylindrical surface, spherical surface or cubic surface accordingly.

3.2. Projective Space Texture

Plenoptic projection functions are used twice in our algorithm. First, as mentioned above, they are used to control the origin and the direction of the tracing ray while rendering the intermediate template. Second, they are used to inversely project the texture in the morphed texture image onto the volume object. That is, they are employed as pseudo-procedure functional models of solid textures. The forward projection (from a volume object to the intermediate template) is through the volume rendering techniques (Direct Volume Rendering, DVR, or, Direct Surface Rendering, DSR [Jon99]). The inverse projection (from the morphed texture image to the voxels in the volume object) is through the projective texture mapping, which can be described as follows:

Assume *p(X,Y,Z)* be a point in the volume object, the centroid position of a plenoptic primitive is (O_x, O_y, O_z), then the texture value at the position *p(X,Y,Z)* could be indexed by a ray passing through *p(X,Y,Z)* and (O_x, O_y, O_z). This ray intersects the morphed texture image at point *S(u,v)* . The texture value (color information, or, for instance, the perturbation coefficients for bump mapping) in the morphed texture image is then calculated and assigned to the position [Win02] of the point *p(X,Y,Z)*.

Note that solid texturing in the traditional sense is defined analytically - by a three-dimensional function (denoted as procedural solid texture) or by discrete samples in the three-dimensional space (denoted as 3D texture or solid texture). Therefore, our technique is more related to environment mapping (also using spherical, cylindrical, and cube maps). However, we see that, in our implementation, the projective textures defined over the space is the same as the solid texture map. As shown in Figure 3, the volume object shown in Figure 2(e) is rendered using DVR and DSR [Jon99]. The top and bottom parts are cut off to show the projective space texture defined within the volume object. Figures 3(a) and 3(b) are rendered using Direct Volume Rendering. Figures 3(c) and 3(d) are rendered using Direct Surface Rendering. Figures 3(c) and 3(d) also show that the iso-surfaces within the volume object are rendered and textured.

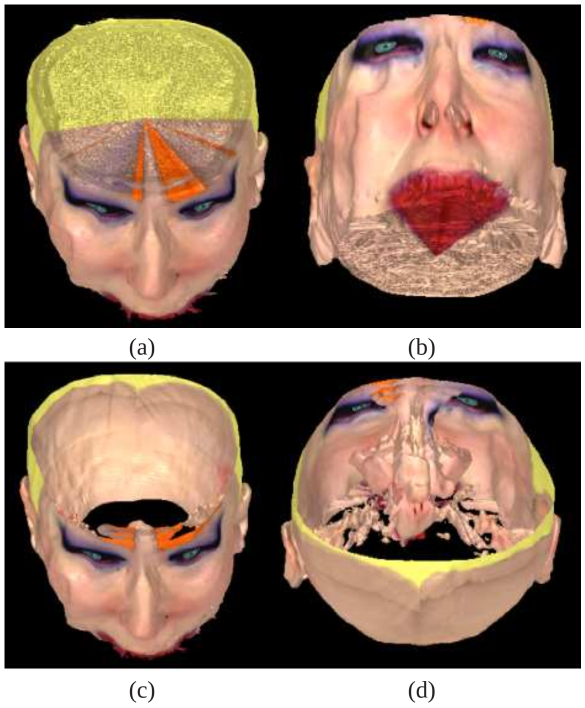


Figure 3: *Projecting a texture image through a volume: The top and bottom parts of the volume object are cut off to show the texture projection within the volume object. Images (a) and (b) are rendered using DVR. Images (c) and (d) are rendered using DSR.*

4. Multi-resolution Representation

It is observed that the continuous representations of the projective space texture could provide a very high-resolution to an intermediate template. However, different areas in the intermediate template may have different resolution requirements for observing the details on the surface of the volume object, for instance, the wrinkles around the eyes of the volume head shown in Figure 2(a). It is impractical and not necessary to render the whole intermediate template using the finest resolution. Therefore, in practice, we use different resolutions to generate different areas in the intermediate template, which we call the multi-resolution representation.

4.1. Multi-resolution Projection of DVR and DSR

As mentioned in Section 2, the parameterization of the plenoptic surfaces could be defined by two continuous variables, $u \in [0,1]$ and $v \in [0,1]$. Therefore, it is possible to generate the intermediate template, in different resolutions, using Direct Volume Rendering (DVR) or Direct Surface Rendering(DSR [Jon99]). The resolution intervals of the intermediate template are defined as $\Delta u = 1/(x_d - 1)$ and $\Delta v = 1/(y_d - 1)$. Then the pixel at the position (u,v) could be indexed by the integral coordinate pair (x,y), $u = x * \Delta u$, $v = y * \Delta v$. The continuous variables u and v are used to adjust the origin and the direction of the tracing ray in the forward projection, thus different resolution intervals could be used to generate the intermediate template.

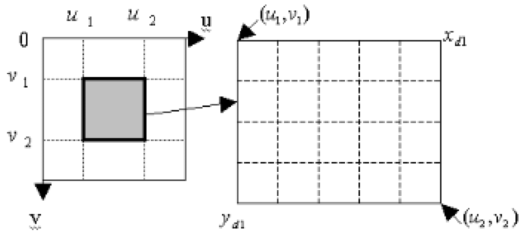


Figure 4: *Multi-resolution Projections.*

In addition, a different high-resolution representation of a rectangular area in the template image could be generated. As shown in Figure 4, (u_1,v_1) and (u_2,v_2) are two different positions in the intermediate template, such that $u_1 < u_2$, $v_1 < v_2$. Then, with new higher resolution intervals, $\Delta u_1 = 1/(x_{d1} - 1)$ and $\Delta v_1 = 1/(y_{d1} - 1)$, the new origin of the tracing ray is:

$$Origin(u,v) = (u_1 + x_1 * (u_2 - u_1) * \triangle u_1, v_1 + y_1 * (v_2 - v_1) * \triangle v_1) \quad (1)$$

where x_{d1} and y_{d1} are the dimensions of the new high- resolution image of the region, and x_1 and y_1 are the integral coordinates, $0 \le x \le x_{d1} - 1$, $0 \le y \le y_{d1} - 1$.

The multi-resolution representation of intermediate template provides the possibilities for the high quality composition of texture images in 3D space, as demonstrated by Froumentin and Willis in their 2.5D rendering and compositing system [FW99]. The multi-resolution intermediate template and the morphed texture image, as well as the hight resolution patch which we will explain later, could be composited and projected onto a volume object directly.

4.2. High Resolution Patch

During rendering, textures can be assigned to any position within the volume object by using a mathematical indexing function, i.e., the inverse of a plenoptic function. That is, the plenoptic models of projective space texture mapping set up the connection between the 3D position in volume object and its 2D projective position on the plenoptic surface. Meanwhile, multi-representations of intermediate templates set up the connection between the 2D projective position (normalized) on the plenoptic surface and its integral coordinates in the morphed texture image. It is obvious that the higher the resolution of the morphed texture image, the higher accuracy of the projective space texture we could get. However, in most cases, we can only get the high-resolution representation of a portion of the morphed texture image, for instance, the close up view of a special area. We call this high-resolution image the *high-resolution patch*.

We assume the high-resolution patch is positioned at a rectangular area in the morphed texture image, which is represented by the coordinates of its two vertices, (u_{a1}, v_{a1}) and (u_{a2}, v_{a2}), where $u_{a1} < u_{a2}$, $v_{a1} < v_{a2}$. Then the position, (x_a, y_a), in the high resolution patch, can be defined as:

$$(x_a, y_a) = ((u - u_{a1})/\triangle u_a, (v - v_{a1})/\triangle v_a) \quad (2)$$

where $\triangle u_a$ and $\triangle v_a$ are the resolution intervals of the high resolution patch, which are represented by the width and the height of the patch. (u,v) are the coordinates of the position in the morphed texture image.

It should be noted that the above two equations, Equations (1) and (2), are not just the resolution representations of an image. Equation (2) is used for projective space texture mapping during volume rendering. Equation (1) is used for adjusting the origin of tracing ray in DVR or DSR [Jon99], for generating the plenoptic surface. These two equations are different and independent texture indexing functions. They are implemented in the processes of inverse projection and forward projection respectively.

4.3. Pixel Level Data Dependent Interpolation

As shown in Figure 1, the warping and morphing operations implemented in steps (3) and (4) may degrade the quality of the 2D plenoptic projected texture image, especially the quality of the tiny texture features in the image. Therefore pixel level data-dependent interpolation [SW04] [SW03] is adapted in the warping and morphing operations in our algorithms, to improve the quality of the morphed texture image.

We adapt and implement this interpolation technique as a plug-in module in the warping and morphing system MORPHOS provided by Gomes et. al. [GDCV99].

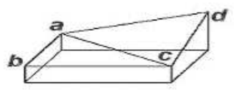


Figure 5: *Pixel level data dependent interpolation: a, b, c, d are four pixel values, represented as heights.*

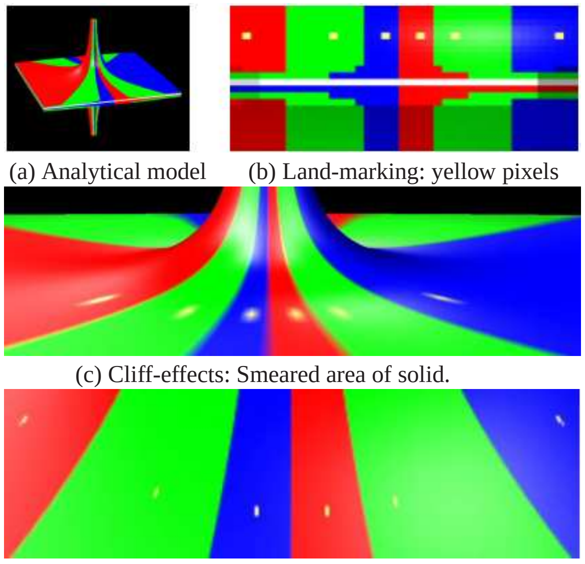
Figure 5 shows four pixel values, represented as heights. Suppose the (low resolution) plenoptic projected image (or a texture photo directly) is X, the high-resolution target image to be generated is Y. First, scan X to initialize a 2D array, x_{mn}, which records the edge direction of all four-pixel squares. Second, scan Y to initialize a 2D array, y_{ij}. Each y_{ij} is inversely mapped to the sample image X, and the array x_{mn} is used to identify the triangle in which the point falls in the source image X. Then the value of y_{ij} is calculated by the use of barycentric interpolation. In the first step, the outlier is detected by comparing the difference $|a-c|$ and the difference $|b-d|$. The pair with smaller difference is treated as an edge. This maintains the smoothness within the regions as well as the sharpness between the flat region and cliff region. With their method, the new high-resolution image has the edges sharp and the smooth areas smooth.

5. Applications and Experiments

In this section, we give the experimental results and address the issues relevant to our contributions. First, we present how a high-resolution patch could eliminate artifacts due to the resolution limitations. Second, we present some applications, which include multi image-based textures mapping and pixel-level data-dependent interpolation.

5.1. Cliff-effect and Anti-alias

An analytical model is built to test the quality of the projective space texture mapping. A typical artefact is the cliff-effect, that is, a single pixel, projected to a surface which is almost parallel to the direction of projection, will be smeared over a relatively large area of solid. It should be noted that even though the procedure solid texture model by definition has infinite detail, the finite resolution of the morphed texture image might still degrade the quality of the rendered image and introduce alias artifacts into the rendered image [HCK*99]. Thus, the implementation of our projective space texture mapping took this into consideration by providing a solution that, a high-resolution texture image as well as a high-resolution intermediate template could eliminate these artifacts reasonably. The idea is similar to the anti-alias method in volume rendering, which fires additional tracing rays to eliminate the alias-artifacts. However, we implement this idea in a reverse sense, that is, in the reverse projection in our algorithms. Figure 6(a) is the constructed analytical model. Figure 6(b) is the morphed texture image. As shown in Figure 6(b), land-marking dots are made in yellow. The size of each marking dot is only a single pixel. Figures 6(c) and 6(d) are the close up views of the textured analytical object. The size of the morphed texture image used in Figure 6(c) is 128x128 pixels, and in Figure 6(d) 1024x1024 pixels. We can see that the cliff effect becomes prominent while the resolution of the morphed texture image is lower, as shown in Figure 6(c). In Figure 6(d), with the benefit of continuous representation of the projective space texture model, as well as the higher resolution of the morphed texture image, the image quality improves greatly.


(a) Analytical model (b) Land-marking: yellow pixels

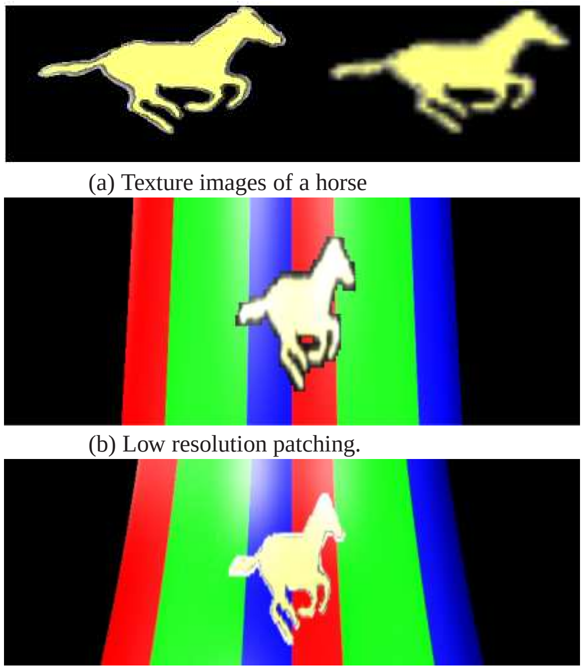
(c) Cliff-effects: Smeared area of solid.

(d) Improved Solid area textured using high resolution patch.

Figure 6: *High resolution projective image: (a) The constructed analytical model. (b) The morphed texture image. Land-marking dots are made in yellow. The size of each marking dot is only a single pixel. (c) and (d) are the close up views of the textured analytical object. The size of the morphed texture image used in figure (c) is 128x128 pixels, and in figure (d) 1024x1024 pixels.*

5.2. High-resolution Patch

In order to overcome the resolution limitations [HCK*99], a high-resolution patch (another higher-resolution image) can be directly imported during texture mapping (with or without morphing). As shown in Figure 7, we would like to paint a horse onto the cliff. The size of the paint area in a low resolution texture image is only 66x20 pixels, however, the size of the horse image is 256x166 pixels. Figures 7 (b)-(c) show the close up views of the paint on the cliff after texture mapping. We can see that the details are preserved quite well in Figure 7(c). However, without high-resolution patching, Figure 7(b) suffers from alias artifacts and the loss of the detail of the horse. Please note that the differences between the areas around the horses' legs in Figure 7(b) and Figure 7(c) are prominent. As previously mentioned, the projective space texture as well as the multi-resolution projections of DVR and DSR are defined in continuous 3D space. Therefore, the inverse projection could be fine-tuned to an extremely small area to digitally composite the texture by using different texture images. As can be seen in Figure 7(c), the textures of the cliff could be successfully composited with the textures of the horse's legs (high resolution image), whereas, in Figure 7(b), the textures of the cliff are completely covered by the textures of the horse's legs (low resolution image).


(a) Texture images of a horse

(b) Low resolution patching.

(c) Improved solid area using high-resolution patch.

Figure 7: *High-resolution patch: (a) the image of cliff-horse, high resolution patch: 256 x 166 pixels (left), low resolution patch: 50 x 32pixels (right, enlarged for display only). (b) Close up view of textured cliff-horse with low-resolution patch (right horse). (c) Close-up view of textured cliff-horse with high-resolution patch (left horse).*

5.3. Multi image-based texture mapping

A spatial transfer function [IDSC04] can be used to control the position of texture mapping. As shown in Figure 8(a), the top half and the bottom half of the head are textured by the use of two different texture images independently. Figure 8(b) shows an example of a concave volume object's texture mapping. The surface of the earth is textured by a spherical texture image. The core of the earth is textured by a cylindrical texture image. It should be noted that, by using an appropriate spatial transfer function [IDSC04], we can let the desired voxel/position be projected into the morphed texture image. Similarly, with the constraints of the spatial transfer function, the color information in the morphed texture image can be only assigned to this voxel/position. This one-to-one indexing mechanism provides the full capability to control the texture positions in either the 2D image or the 3D space during texture mapping.

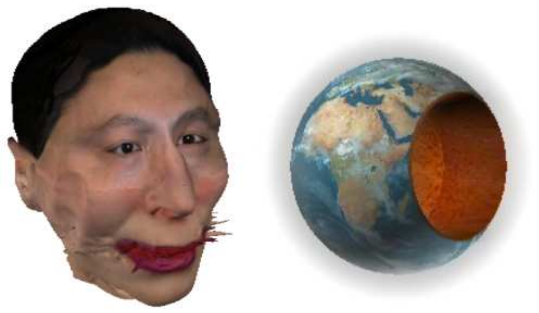
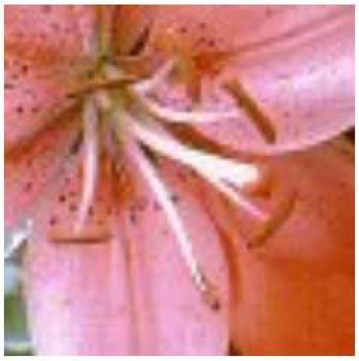

(a) Head (b) Earth

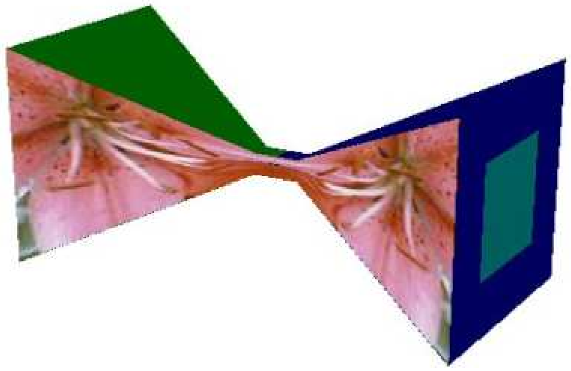
Figure 8: *Multi image-based textures mapping: (a)The top half and the bottom half of the head are textured by the use of two different spherical texture images independently. (b)The surface of the earth is textured by a spherical texture image. The core of the earth is textured by a cylindrical texture image.*

5.4. Pixel level data dependent interpolation

Pixel level data dependent interpolation preserves edge details by keeping edges sharp and smooth areas smooth. Figures 9 and 10 give an example of the application of the interpolation method. For comparison purpose, the bilinear interpolation is also implemented in our algorithms. Figure 9(a) is the original photo of stamens. The image size is 75x75 pixels. Figure 9(b) is the textured volume object. Figure 10(a) is the morphed texture image (2048x2048 pixels). Figure 10(b) shows the close-up views of the rendered image. The stamens on the left side in Figure 9(b), Figure 10(a) and Figure 10(b) (Images I and II) are interpolated by the pixel level data dependent interpolation, the stamens on the right side the bilinear interpolation. The close up views given in Figure 10(b) show that the pixel level data dependent interpolation gives better appearance while details of the textures are preserved quite well. Figure 11 shows an enlarged image of the textured object, which is shown in Figure 9(b).



(a) Stamens



(b) Textured volume object.

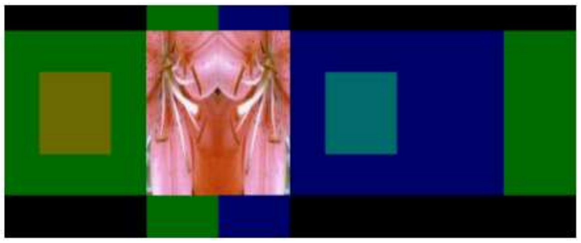
Figure 9: *Pixel level data dependent interpolation: (a)The original photo of stamens, 75 x 75 pixels. (b)Textured volume object. The stamens on the left side are interpolated by the pixel level data dependent interpolation, the stamens on the right side the bilinear interpolation.*

6. Conclusions

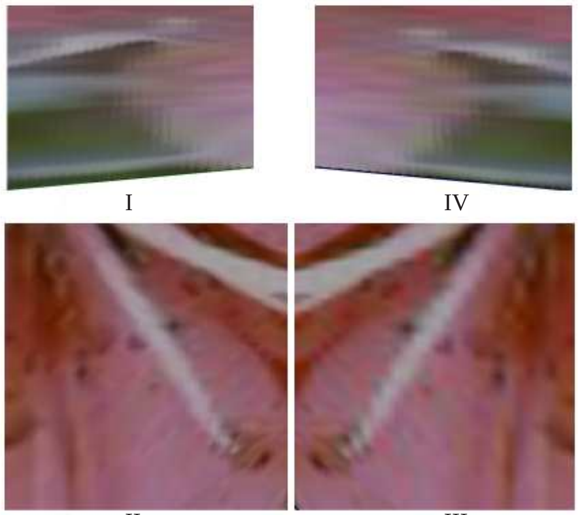
Multi-resolution representations of image-based two-part texture mapping for volume objects have several advantages. First, the multi-resolution representation of the intermediate template gives us a microscope to observe/texture the tiny structures of the surfaces of a volume object, for instance, the wrinkles around the eyes of a volume-based animation character. Second, as projecting 2D texture to 3D voxels is an ill-conditioned problem, we are trying to find a reasonable method to handle the "fall-over" pixels, while generating the intermediate template (forward projection); and we are also trying to handle the voxels in the concave potions of the volume object during texture mapping (inverse projection). Note that our methods could be directly hooked onto volume rendering engine DSR and DVR, therefore the solution to these two problems is given by using spatial transfer constraints and semantic constraints [IDSC04].

7. Further Works

Using the high-resolution morphometric registration method presented by Shen and Davatzikos [SD03] and appropriate spatial transfer constrains [IDSC04], our method could possibly be used for a volume object's segmentation and registration. In addition, since we base our work on DSORs [CIJ*05], our methods could be applied to "Point-based" techniques in computer graphics [KB04]. As Kobbelt and Botsch pointed out recently in their survey [KB04], the critical issues of the point-based techniques are multi-resolution representations, pixel/voxel based surfaces/disks, manifold conditions, point-based rendering, etc. Therefore, our new challenges will be seeking reasonable models for intermixing point-sampled models (not discretely sampled voxels/pixels) and image-based textures. Other further works will be focused on constructing semantic-constraints to solve self-occlusion problems during texture mapping.



(a) The warped texture image.



(b) Close-ups of Figure 9(b).

Figure 10: *The morphed texture image(cubic projection) and close up views: (a)The warped texture image: the size of the image of stamens is 257x682 pixels. Left: pixel level data dependent interpolation, right: bilinear interpolation. (b)Close up views in Figure 9(b): Images I and II: pixel level data dependent interpolation; Images III and IV: bilinear-interpolation.*

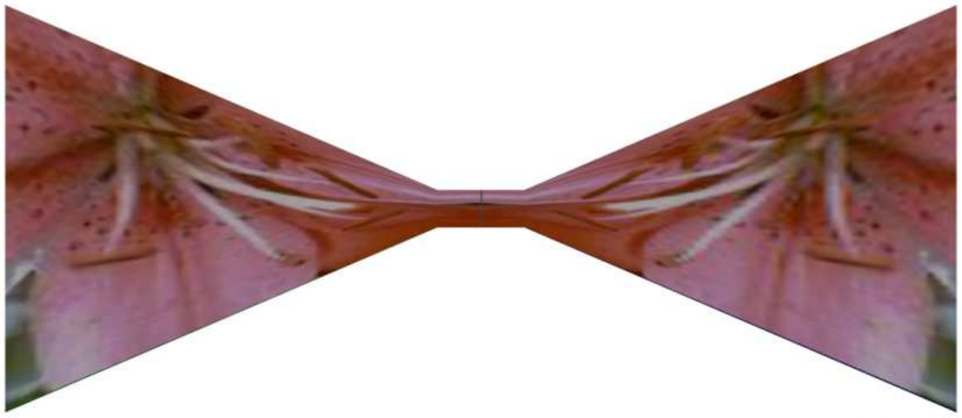


Figure 11: *The rendered image of textured volume object. Left stamens: pixel level data dependent interpolation; Right: bilinear interpolation. The deformation of the volume object is through spatial transfer function [IDSC04]. The volume object is shown in Figure 9(b).*

References

[AW96] ALLEBACH J., WONG P.: Edge directed interpolation. In *Proc. IEEE Int. Conf. Image Processing* (1996), vol. 3, pp. 707–710.

[CIJ*05] CHEN M., ISLAM S., JONES M., SHEN P., SILVER D., SINGH V., WALTON S. J., WILLIS P. J.: Deforming and animating discretly sampled object representation. Eurographics 2005, STAR Report, accepted., August 2005.

[ES86] E.A.BIER, SLOAN K.: Two-part texture mapping. *IEEE Computer Graphics and Application 6*, 9 (1986), 40–53.

[FW99] FROUMENTIN M., WILLIS P.: An efficient 2.5d rendering and compositing system. *Computer Graphics Forum 18*, 3 (1999), 385–394.

[GDCV99] GOMES J., DARSA L., COSTA B., VELHO L.: *Warping and Morphing of Graphical Objects*, first ed. Morgan Kaufmann Publishers, 1999.

[HCK*99] HART J. C., CARR N., KAMEYA M., TIBBITS S., COLEMEN J.: Antialiased parameterized solid texturing simplified for comsumer-level hardware implementation. In *SIGGRAPH/Eurographics Workshops on Graphics Hardware* (Aug. 1999), pp. 45–53.

[IDSC04] ISLAM S., DIPANKAR S., SILVER D., CHEN M.: Temporal and spatial splitting of scalar fields in volume graphics. In *Proc. IEEE VolVis2004* (Oct. 2004), pp. 87–94.

[Jon99] JONES M. W.: Direct surface rendering of general and genetically bred implicit surfaces. In *In Proc. 17th Ann. Conf. of Eurographics (UK Chapter)* (1999), pp. 37–46.

[KB04] KOBBELT L., BOTSCH M.: A survey of point-based techniques in computer graphics. *Computer and Graphics 28*, 6 (Dec. 2004), 801–814.

[LO00] LI X., ORCHARD M.: New edge directed interpolation. In *Proc. IEEE Int. Conf. Image Processing* (2000), vol. 2, pp. 311–314.

[Pea85] PEACHY D.: Solid texturing of complex surfaces. *Computer Graphics (Proceedings of SIGGRAPH 85) 19*, 3 (1985), 279–286.

[Per85] PERLIN K.: An image synthesizer. *Computer Graphics, (Proceedings of SIGGRAPH 85) 19*, 3 (1985), 287–296.

[PH89] PERLIN K., HOERT E.: Hypertexture. *Computer Graphics 23*, 3 (1989), 253–262.

[SD03] SHEN D., DAVATZIKOS C.: Very high resolution morphomety using mass-preserving deformation and hammer elastic registration. *NeuroImage 18*, 1 (Jan. 2003), 28–41.

[SJ01] SATHERLEY R., JONES M. W.: Hypertexturing complex volume objects. In *The 9-th International Conf. in Central Europe on Computer Graphics, Visualization and Computer Vision 2001* (2001), pp. 146–153.

[SW03] SU D., WILLIS P.: Demosaicing of color images using pixel level data-depedent triangulation. In *Eurographics UK Conferences, Theory and Practice of Computer Science* (2003), pp. 16–23.

[SW04] SU D., WILLIS P.: Image interpolation by pixel level data-dependent triangulation. *Computer Graphics Forum 23*, 2 (2004), 189–201.

[Win02] WINTER A. S.: *Volume Graphics:Field-Based Modelling and Rendering*. PhD thesis, University of Wales at Swansea, 2002.