EUROGRAPHICS 2002 / G. Drettakis and H.-P. Seidel
(Guest Editors)

Volume 21 (2002), Number 3

Projective Texture Mapping with Full Panorama

Dongho Kim and James K. Hahn

Department of Computer Science, The George Washington University, Washington, DC, USA

Abstract

Projective texture mapping is used to project a texture map onto scene geometry. It has been used in many applications, since it eliminates the assignment of fixed texture coordinates and provides a good method of representing synthetic images or photographs in image-based rendering. But conventional projective texture mapping has limitations in the field of view and the degree of navigation because only simple rectangular texture maps can be used.

In this work, we propose the concept of panoramic projective texture mapping (PPTM). It projects cubic or cylindrical panorama onto the scene geometry. With this scheme, any polygonal geometry can receive the projection of a panoramic texture map, without using fixed texture coordinates or modeling many projective texture mapping. For fast real-time rendering, a hardware-based rendering method is also presented. Applications of PPTM include panorama viewer similar to QuicktimeVR and navigation in the panoramic scene, which can be created by image-based modeling techniques.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Viewing Algorithms; I.3.7 [Computer Graphics]: Color, Shading, Shadowing, and Texture

1. Introduction

Texture mapping has been used for a long time in computer graphics imagery, because it provides much visual detail without complex models[7]. A texture map is generally a two-dimensional image, and accessed using texture coordinates assigned to the geometry of scenes. Although color is the content of texture maps in most cases, texture elements (texels) can have other characteristics, such as transparency, bump perturbation, and normal vectors. Recently, texture mapping has become more important, thanks to the enhancement of graphics hardware. It is a key component in real-time graphics applications such as games or virtual reality.

Projective texture mapping is a relatively new technique of mapping textures. Rather than assigning fixed texture coordinates to geometry, projective texture mapping projects a texture map onto geometry, like a slide projector. Projective texture mapping is more convenient to use in many applications than assigning fixed texture coordinates. For example, light mapping, which enables fast complex light contributions such as Phong shading or spotlight, can be implemented easily with projective texture mapping. Image-based rendering (IBR) draws more applications of projective texture mapping. When photographs or synthetic images are used for IBR, those images contain scene information, which is projected onto the screen in the capturing process. Therefore, projecting them into scenes can be used to model image-based virtual environments.

However, projective texture mapping has some limitations. It can project only one rectangular texture map using one perspective frustum. Therefore, the field of view determined by the projection is limited. In many image-based applications, we must consider the boundaries of projection carefully and two or more projectors should be used as needed. Moreover, cylindrical or cubic panorama cannot be used as projecting sources.

In this work, we propose the concept and algorithms for the projection of panoramic images. This means that 360° panorama can be projected to virtual environments directly. Therefore, many image-based applications can handle the projections in a more convenient way. As panoramic images, we use cubic panorama and cylindrical panorama. Cubic panorama consists of six square faces,

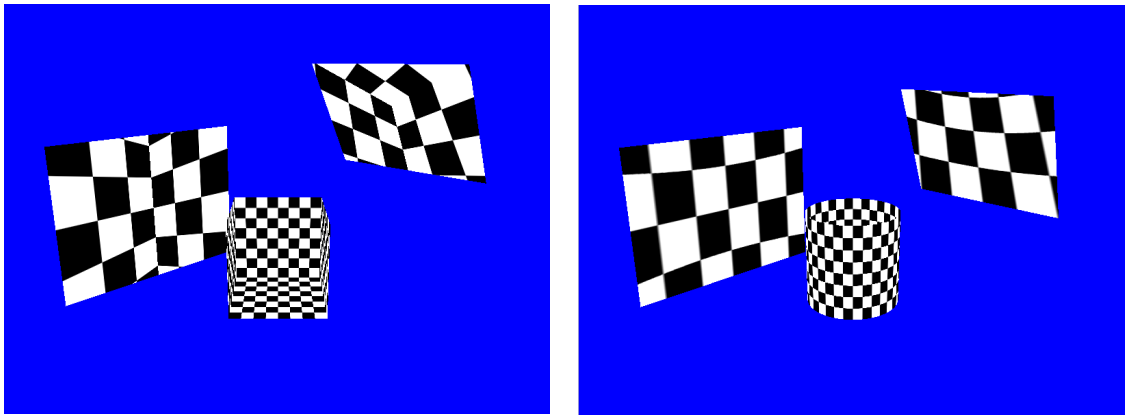


Figure 1: Panoramic projective texture mapping
Left image is the projection from cubic panorama (two faces of the projection source are not shown). Right image is the projection from cylindrical panorama.

which capture what is visible along all directions from a viewer position. A cylindrical panorama is a rectangular texture map, which contains texels parameterized with cylindrical coordinates. Figure 1 shows the concepts of projective texture mapping from panorama. Our method achieves real-time performance, because it is implemented fully by 3D graphics hardware with DirectX 8 API. From now on, we will refer to our method as panoramic projective texture mapping (PPTM) to distinguish it from standard projective texture mapping.

2. Related Work

2.1. Texture mapping and Environment mapping

Texture mapping is a technique of introducing visual details into a scene[7]. There are many variations in texture mapping, mainly according to the dimensionality and usage. 1D, 2D, or 3D textures can be used according to the dimensions of the data domain. Texture maps can represent surface colors, transparency, bumps, or normal vectors according to the contents of the map.

Texture maps may contain the environment to be used for environment mapping[6]. In environment mapping, what is visible from a point is captured and contained in a texture map. It is accessed with the direction of reflected view vector, so that surfaces can reflect the environment. Ray-tracing-like reflection can be performed easily without complex computation. Recent 3D graphics hardware has built-in environment mapping module, so that it can be performed in real-time.

2.2. Projective texture mapping and its hardware implementation

Texture mapping is performed by the assignment of texture coordinates to all the points on the scene geometry. Usually, texture coordinates are assigned as parametric coordinates of surfaces or assigned manually. But projective texture mapping provides another method of texture coordinate assignment. With this method, a texture is projected onto object surfaces, like slide projection[15].

Most of the current graphics hardwares can handle projective texturing[1,11]. First, all the points on the scene geometry are given 3D texture coordinates which are their camera space locations. These coordinates are then transformed to the projection coordinate system, where the origin is at the center of projection. The coordinates are divided by z-component in order to perform perspective projection. This procedure is similar to the viewing and projection transformations in normal image rendering, and can be represented as a single 4×4 matrix. This matrix is set as a texture transformation matrix maintained by the graphics hardware. When polygons are drawn with this configuration, the automatic transformation of texture coordinates gives the final 2D coordinates to access the texture map.

Projective texture mapping is useful in many image-based applications. If the images are taken from photographs of real scenes, they can be thought of as transformed perspectively in the capturing process. Therefore, virtual environment can be modeled easily using the images as projecting sources. Moreover, it eliminates the need for complex or laborious process of texture coordinate assignment.

However, the conventional projective texture mapping technique is limited to one rectangular image. Therefore, projective texturing cannot be performed with panoramic images. Moreover, with the limited field of view, the coverage of the projection is limited only to a part of the scene, which may be a problem in many VR applications.

2.3. Panoramic rendering from fixed viewpoint

QuicktimeVR[3] introduced a simple, but realistic virtual reality system. It uses cylindrical images and provides look-around views from a fixed viewpoint. Even though viewpoints cannot be relocated smoothly, QuicktimeVR has been used widely because it can generate realistic virtual environments easily from photographs. There are many commercial software packages for authoring and rendering cylindrical panorama[16]. Most of them employ software-based renderers, since rendering cylindrical maps involves nonlinear warping and is difficult to handle with graphics hardware.

Cylindrical panorama can be rendered in hardware by modeling polygonal approximation and texture pre-warping. In other words, arbitrary polygonal models can be used in the panorama authoring stage, so that the model contains texture maps, which are pre-warped appropriately[14].

2.4. Image-based modeling using a single image

In image-based modeling, 3D scene modeling is performed based on input images. The types and sizes of modeling entities are extracted from images, usually with the user's intervention, and input images are used as texture maps with the extracted geometry[4,8,9,10,12]. The objective of these approaches is modeling a 3D scene so that it can represent the contents in source images as well as possible.

Some of the approaches use multiple images[4] or a video sequence. On the other hand, a single image can also be used for image-based modeling. Tour-into-the-picture (TIP) [8] is one of the approaches. With this scheme, a simple rectangular structure is constructed by the user's specification of the vanishing point. The source image is then used as a texture map of the rectangles for future navigation. For rendering, it uses its own logic implemented in software, to determine texture coordinates for the points on the geometry.

TIP was extended by Kang et al.[9], so that broader range of images can be handled by using vanishing lines. Spherical and cylindrical panorama can also be used as source images. It is implemented by modeling the background as a large spherical or cylindrical mesh with the source panorama as textures. OpenGL performs real-time rendering. In order to use panoramic images without warping, the background mesh should be tessellated into many polygons. Since the panorama is not projected to the scene, panoramic-to-planar pre-warping is needed for foreground objects in order to extract textures for them.

All of these works use software rendering or non-projective texture mapping, which assigns static texture coordinates to the polygonal mesh. Software rendering is slow. Moreover, non-projective texture mapping requires warping of the source image in order to nullify the effect of projection during the capture of the source image. Using projective texture mapping would simplify the modeling and rendering processes.

3. Panoramic projective texture mapping (PPTM)

Conventional projective texture mapping uses simple planar image as its source of projection. However, its field of view is limited, and the degree of freedom in virtual environment modeling is also limited. Panoramic images, on the other hand, provide wider fields of view for the projection. Therefore, the whole environment can receive the projection from one image.

We use two types of panorama as projection sources, cubic and cylindrical. The details of these parameterizations are explained in the next two sections. With these schemes, virtual environments can be represented by rendering polygonal models with PPTM enabled. An efficient implementation will also be presented, which takes advantage of hardware acceleration.

4. Projective texturing from cubic panorama (Cubic PPTM)

Cubic panorama captures 360° field of view horizontally and vertically, and stores the result in six faces of a cube.

With conventional projective texture mapping, cubic panorama can be projected onto scene geometry with six concurrent projective texturing, each with perspective projection of 90° field of view. But projection of cubic panorama is not simple when two or more textures out of the six are projected onto a polygon simultaneously, because hardware texturing requires that a texture be assigned to a whole polygon. The images in Figure 1 show these cases. In these cases, the polygon should be subdivided along the projection of frustum boundaries, so that each can have its own texture. But this may make the scene more complex. Moreover, dynamically changing

texture mapping cannot be handled, such as moving objects in static projection or moving projection source. Therefore, we propose a method to project from a cubic texture directly to overcome these limitations.

Recent graphics hardware provides cubic environment mapping as a method of environment mapping. Texels in cubic panorama are indexed with the directional 3-vector from the center of the cube. So, reflection on a point is modeled as a texel indexed by the reflected view vector at the point.

Instead of using it for reflection, we can use the cubic environment mapping method for projective texture mapping with cubic panorama. A 3D point receiving the projection is transformed into the coordinates in projection space, where the origin is at the center of the cubic projection and the axes are aligned with the cube. Then, the transformed coordinates can be used to index the cubic panorama with cubic environment mapping hardware. In this way, any polygon can receive projection from cubic panorama.

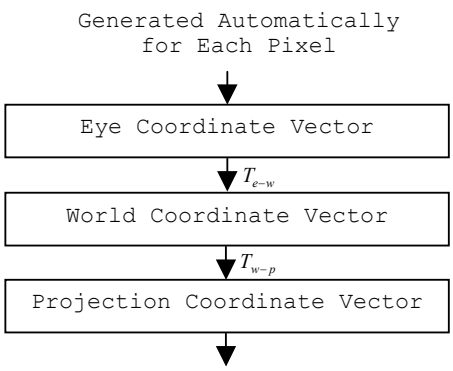


Figure 2: Determination of Texture Coordinates for Cubic PPTM

Texture coordinates for cubic PPTM are determined as shown in Figure 2. Graphics hardware can generate 3D texture coordinates with camera space positions automatically. Let the eye coordinate vector be (x, y, z) and T_{e-w} be the transformation from eye coordinates to world coordinates. T_{e-w} is the inverse of the viewing transformation, which can be retrieved using DirectX API. T_{w-p} is the transformation from world coordinates to projection coordinates. The projection coordinate system has the origin at the center of the cube and the axes are aligned with the cube. T_{w-p} is determined by placing and orienting the projection source. Then, $T = T_{w-p}T_{e-w}$ gives the whole transformation from eye coordinates to projection coordinates. If T is set as the texture transformation matrix, graphics hardware transforms the texture coordinates generated automatically with this transformation. With these configurations, texture mapping with cubic projection is performed by setting the texture operation mode to cubic environment mapping. All of these operations are performed on a per-pixel basis by the graphics hardware. Conventional projective texture mapping is implemented in a similar way. The only difference is that it performs division-by-z in projection coordinates to get 2D texture coordinates, while our cubic projection uses the 3D vector in projection space in order to determine the texture coordinates for cubic environment mapping.

The pseudo code for rendering is shown in Figure 3. DirectX API does all configuration and rendering. There is much flexibility in the assignment of textures, i.e., two or more projections can be performed simultaneously and some polygons may not receive any projection.

```
T : Texture Transform Matrix
T_{e-w} : Inverse of Viewing Transform
T_{w-p} : Transformation
        from World Coordinates
        to Projection Coordinates

for each frame
   Configure Automatic Texture
       Coordinate Generation Mode
       as Camera_Space_Position
   Configure Cubic Environment
       Mapping mode
   Compute T_{e-w}
   for each Cubic Projective Texture
      Set texture
      T = T_{w-p} T_{e-w}
      Set Texture Transformation
          Matrix as T
      Draw Polygons receiving
          Current Projection
   endfor

   Configure Rendering mode
       to normal rendering
   Draw Polygons Not receiving
       Projective Texture
endfor
```

Figure 3: Pseudo code of Cubic PPTM

5. Projective texturing from cylindrical panorama (Cylindrical PPTM)

Cylindrical textures are being used in many applications, since they provide a 360-degree horizontal field of view and constant horizontal sampling density. However, using cylindrical maps directly with hardware acceleration is difficult, because cylindrical mapping involves nonlinear computations, which cannot be handled easily by graphics hardware. So, in many look-around applications with cylindrical maps, such as QuickTimeVR, texture coordinates are determined by the software renderer.

Since using cubic panorama is usually faster than using cylindrical panorama, a cylindrical map may be resampled into a cubic map. But cylindrical maps are easy to obtain and provide constant horizontal sampling rates. Moreover, resampling may degrade the quality of the maps.

Projection with cylindrical panorama is a process of warping a cylindrical texture map onto a plane. We propose an algorithm for this process using graphics hardware.

5.1. Main idea

First, we define *projection plane*, onto which a cylindrical map is warped locally. Then, conventional planar projective texture mapping is used with the warped texture map on the *projection plane*. In order to utilize graphics hardware and enhance rendering performance, we perform the nonlinear local warping on-the-fly using bump mapping hardware.

From now on, we use (θ, v) for the cylindrical texture coordinates in [0..1] for cylindrical panorama, and (u', v') for the planar texture coordinates on the projection plane. 3-vector (x, y, z) is a location in projection coordinate system, where the origin is at the center of the cylinder. The *z*-axis is aligned with the axis of the cylinder, and the *x*-axis is aligned with $\theta = 0$. Transformation from eye coordinates to projection coordinates is performed in the same way as cubic PPTM.

Let H be the height of the unit cylinder given as $\mathrm{H} = 2\tan(0.5 \times \mathrm{FOV})$, where FOV is the vertical field of view of the cylindrical panorama. Then, (θ, v) is related to (x, y, z) as follows. Here, atan2() is arctangent function in standard C library, which gives angle in [-π..π] for output.

$$\theta = \frac{\mathrm{atan2}(y, x)}{2\pi} + 0.5$$
$$v = \frac{1}{\mathrm{H}} \frac{z}{\sqrt{\mathrm{x}^2 + \mathrm{y}^2}} + 0.5 \qquad (1)$$

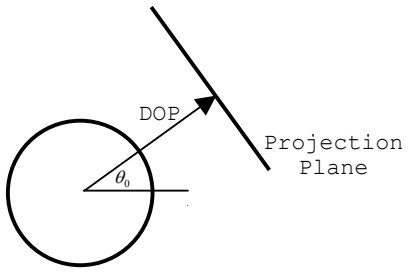


Figure 4: Projection Plane and Direction of Projection (DOP)

For local warping, we define the *direction of projection (DOP)*, which becomes the normal vector of the projection plane. In other words, DOP is the direction that we are most interested in for the projection. We use DOP perpendicular to the cylinder, so it is defined simply by θ_0, which is the angle of DOP from the *x*-axis. Then, the projection plane is placed at unit distance from the origin with the normal vector along DOP. Figure 4 shows a 2-dimensional view of this configuration.

Suppose that (x', y') is a projection of (x, y, z) onto the local coordinates of the projection plane. Then, the above equations become the following equations which determine (θ, v). And it determines (θ, v) for given (x', y').

$$\theta = \theta_0 + \frac{\mathrm{atan}(x')}{2\pi}$$
$$v = \frac{1}{\mathrm{H}} \frac{y'}{\sqrt{1 + \mathrm{x}'^2}} + 0.5 \qquad (2)$$

The main idea of our work is to decompose these equations into a linear approximation and nonlinear residuals, and fill the bump map with the nonlinear part. The bump map is used to perturb texture coordinates determined by the linear part. This is done by decomposing Equation 2 as follows.

$$\theta = \theta_0 + ax' + R_\theta(x')$$
$$v = 0.5 + by' + R_v(x', y') \qquad (3)$$

The residual parts are given as follows.

$$R_\theta(x') = \frac{\mathrm{atan}(x')}{2\pi} - ax'$$
$$R_v(x', y') = \frac{1}{\mathrm{H}} \frac{y'}{\sqrt{1 + \mathrm{x}'^2}} - by' \qquad (4)$$

Here, a and b are the coefficients of the linear approximations of (θ, v) according to (x', y'). They are computed before rendering so that the magnitudes of the residual parts are minimized. Because texture coordinates are interpolated linearly by texture mapping hardware, we encode the linear components as regular texture coordinates of the polygon. The residual components are encoded as bump maps to perturb the linear texture coordinates. Implementation details are explained in the next subsection.

5.2. Using environment mapped bump mapping (EMBM) for hardware support

Recent improvement in texture mapping hardware introduced multi-texture mapping. With multi-texture mapping, two or more textures can be applied to a polygon in a single rendering pass. Each texture is applied at each stage. The result of the previous stage and the texture for the current stage are combined by various operations, such as add, subtract, multiply, etc.

One of the operational modes of multi-texturing is environment mapped bump mapping (EMBM) where texture coordinates in the second stage are perturbed by the texel values sampled in the first stage. This functionality is designed originally for bumped reflection, where the environment is reflected using environment mapping. In this work, however, this functionality is used to perturb texture coordinates and perform nonlinear warping with hardware support. Because residual parts shown in Equation 4 are approximated as a bump perturbation map, the resolution of the bump map may affect the quality of projective texturing. 256×256 bump map is used in this work, and there is no noticeable artifact due to the approximation.

5.2.1. Implementation with single-pass rendering

Cylindrical PPTM can be implemented using EMBM and multi-texturing capabilities of 3D graphics hardware. With multi-texture capability, multiple textures can be applied onto object geometry in a single rendering pass. Figure 5 shows the procedure for determining texture coordinates. For the first texture stage, texture operation is set as EMBM and bump perturbation map is used as texture. The bump perturbation map contains the amounts of perturbation needed for all texels, which are computed by the residual parts of cylindrical parameterization given in Equation 4. For the second texture stage, the cylindrical

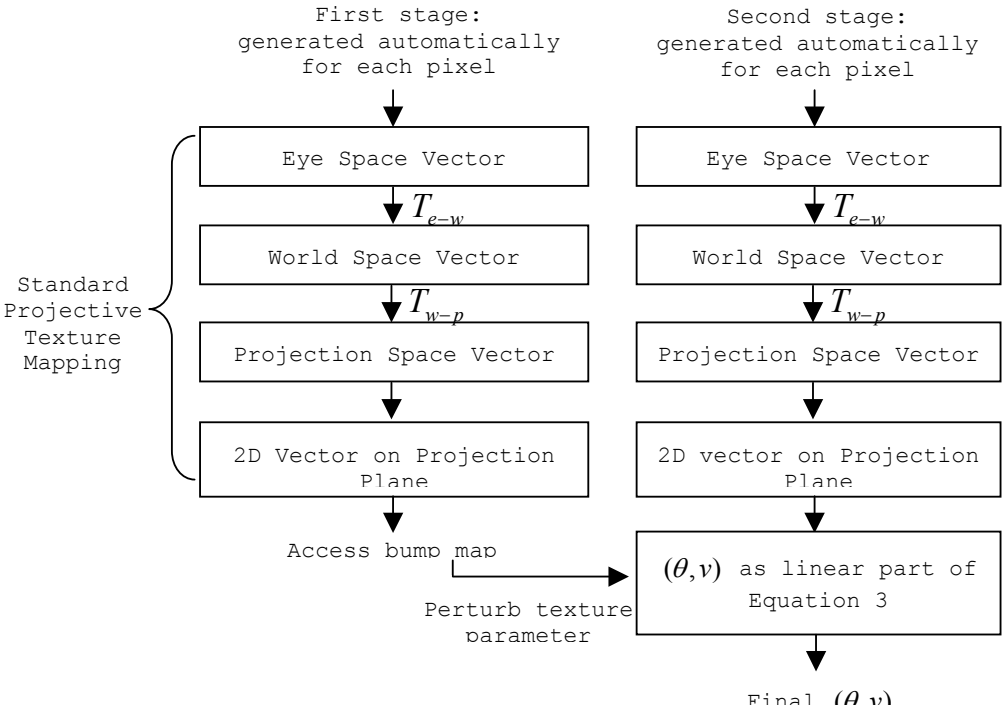
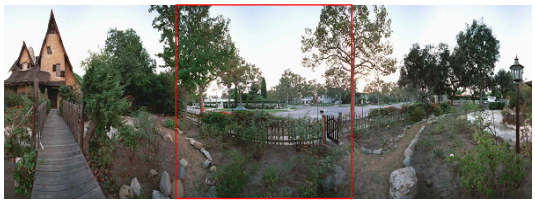


Figure 5: Determination of texture coordinates for cylindrical PPTM



(a) Cylindrical map

(b) Projected to Projection Plane

(c) Final Result

Figure 6 Cylindrical PPTM with two-pass rendering: The area in red rectangle in (a) is specified by DOP and it is projected using EMBM (first pass). Standard projective texturing with (b) results in final result in (c).

panorama is used as the texture. For both stages, texture coordinates are generated by automatic texture generation capability, and transformed appropriately by the texture transformation matrix. Most of the procedure is similar to standard projective texture mapping. But the use of EMBM makes it possible to perform nonlinear warping with graphics hardware.

We found that current 3D graphics hardware has problems in implementing this procedure, even though DirectX supports the capability. We tested nVIDIA GeForce3 and ATI Radeon. They do not function correctly when EMBM is used and projective texturing is turned on in multi-stages. However, DirectX reference renderer (software renderer) executes this implementation correctly. It seems that current hardware did not implement the entire functionality because this usage is beyond the scope of its original objective, which is environment mapping. We hope that full functionality is supported in the near future.

Therefore, we propose another rendering method in the next subsection, which is less efficient but possible with current graphics hardware.

5.2.2. Implementation with multi-pass rendering

Since many graphics hardware do not function correctly when EMBM and projective texturing are used simultaneously, we propose a multi-pass rendering approach here. It is similar to the single-pass approach in the previous subsection, but it renders in two passes, i.e., rendering takes place two times for a final result. In the first pass, texture operation mode and texture maps are set up in the same way as single-pass implementation. But projective texturing is not set up, and viewing direction is aligned with the direction of projection (DOP). This results in the image, which contains a part of the panorama projected onto the projection plane. In the second rendering pass, this image is used as texture and appropriate projective texturing is set up for the final image. Figure 6 shows the images generated in this process.

Because we warp the cylindrical map explicitly, rendering performance with this method is lower than the single-pass method. Moreover, it is required to perform the first pass rendering more than once, since local warping cannot cover all views from the projection source. Polygons can be grouped according to their locations, so that each group can be projected with each first-pass rendering result. This problem does not occur with the single-pass method.

6. Applications and Experimental Results

Cubic and cylindrical PPTM were implemented using DirectX 8 API. Similar implementation would be possible with OpenGL. In the following subsections, various applications of PPTM and their results are presented.

6.1. Panorama Viewer (viewpoint and projection source at the same location)

When the source of projection is located at the viewpoint, PPTM works as panorama viewer, which is similar to QuicktimeVR. Since panorama is projected from the viewpoint and rendered from the same location, we can use any scene geometry as long as it covers the entire screen. Because it is the simplest geometry, we use a rectangle placed on the screen of the viewing frustum. In other words, the rectangle is rendered from the viewpoint, while it receives the projection from the viewpoint. This rectangle is always attached to the viewing frustum when the user rotates the viewing direction or changes the field of view. Our panorama viewer supports both cubic and cylindrical panorama, and Figures 7 and 8 show the rendered results using the panorama viewer.

With the cubic panorama viewer, everything visible from a fixed viewpoint can be captured and looked around. Figure 7(a) is the cubic panorama data we used. Figure 7(b) shows rendered images when the user changes viewing direction of field of view. This functionality can be obtained by rendering six squares with static texture coordinates. However, we believe that the approach of projection gives possibility for more applications.

In the cylindrical panorama viewer, the direction of projection (DOP) changes according to the viewing direction. Cylindrical panoramic viewer provides the same rendering capability as QuicktimeVR. But our viewer renders fast even with large resolution, because it takes advantage of hardware rendering. On the other hand, software renderer such as QuicktimeVR degrades image quality when the user changes viewing direction in order to achieve real-time functionality. Figure 8(a) is a sample panorama and Figure 8(b) shows various rendered images.

6.2. Projection of Panorama onto Geometry

In Figures 9 and 10, panoramas are projected onto polygonal geometry. In our examples, panoramas rotate while they are projecting the textures, which is not possible with static texture mapping.

In the results of Figure 9, the cubic panorama in Figure 7(a) was used, and simple cubic geometry in Figure 9(a) is used. In the rendered images in Figure 9(b), the colors of the polygons are modulated by the projected texture colors. It is shown that two or three faces of the texture contribute together to one polygon, which is not possible with conventional projective texture mapping. This example is a kind of light mapping, and the modeling of full 360º light is possible to be used for light mapping.

Figure 10 is a similar experiment with cylindrical panorama. In this case, the polygons selects incoming texture colors as their final colors.

6.3. Rendering speed

We used a Pentium III 450 MHz PC with GeForce3 graphics card. For cubic PPTM, rendering speed is very high. When cylindrical panorama is used, the frame rate is much lower than the case of cubic panorama. This is due to the overhead of multi-pass rendering. In the example of cylindrical projection, we performed three renderings for the first pass, because single explicit warping cannot cover all the polygonal models.

However, it still shows high frame rates even for large resolutions, when compared with software renderers. If future graphics hardware works well with single-pass implementation of cylindrical PPTM, the frame rate will increase a lot.

The next table shows the measured rendering frame rates in frames per second.

Resolution	*640 x 480*	*1024 x 768*
CUBIC VIEWER	652	269
CYL VIEWER	140	104
CUBIC PROJECTION	469	227
CYL PROJECTION	53	46

7. Conclusion and future work

We presented the concept of panoramic projective texture mapping (PPTM), and proposed algorithms to use cubic and cylindrical panorama as projecting sources. With this scheme, a single panoramic texture can be projected to the whole scene without additional modeling or subdivision of polygonal models. Examples of the projection were shown for panoramic viewer as well as more general cases. We believe that projective texture mapping from panorama can be used in many useful applications.

As future work, our method could be applied to many image-based rendering techniques. First, it could be used with an image-based modeler so that a panorama can work

as an input image for modeling. Virtual environments with more degrees of navigation could be generated.

Since photographs can be modeled as projecting texture into the scene, many image-based rendering techniques use projective texture mapping. Our method could be combined with some of existing image-based rendering techniques using projective textures[2,4]. Another future work could be the projection of more complex texture maps, which are modeled nonlinearly. For example, concentric mosaics[13] is similar to cylindrical map, but it is parameterized differently to capture horizontal parallax with 3D data. It is not possible to render it using 3D graphics hardware due to its nonlinearity. Since our work proposes the baseline of projecting nonlinear texture map using graphics hardware, it could be extended to using concentric mosaics as rendering entity.

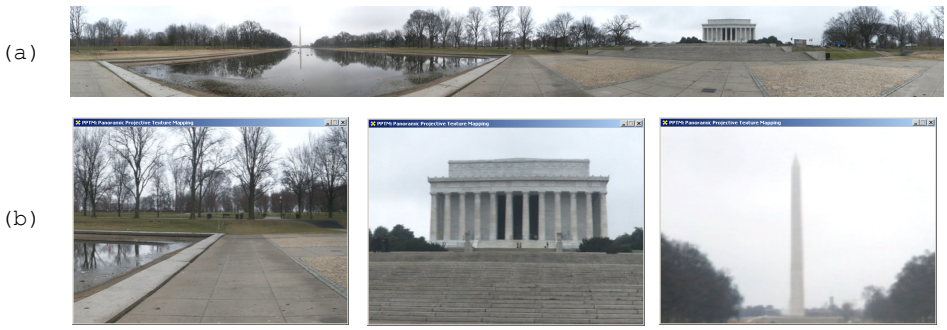
Acknowledgement

The authors wish to thank Ge Jin for the help in the preparation of demo videos. This work was partially supported by George Washington University Presidential Fellowship.

References

1. D. Blythe, et al., "Advanced Graphics Programming Techniques Using OpenGL", *Siggraph 2000 Course Notes*.
2. C. Buehler, M. Bosse, L. McMillan, S. Gortler, and M. Cohen, "Unstructured Lumigraph Rendering", *Proceedings of SIGGRAPH 2001*, 425-432, 2001.
3. S.E. Chen, "Quicktime VR – An Image-Based Approach to Virtual Environment Navigation", *Proceedings of SIGGRAPH 95*, 29-38, 1995.
4. P.E. Debevec, C.J. Taylor, and J. Malik, "Modeling and Rendering Architecture from Photographs: A Hybrid Geometry- and Image-Based Approach", *Proceedings of SIGGRAPH 96*, 11-20, 1996.
5. P.E. Debevec, Y. Yu, and G.D. Borshukov, "Efficient View-Dependent Image-Based Rendering with Projective Texture-Mapping", *Eurographics Rendering Workshop 1998*, 105-116, 1998.
6. N. Greene, "Environment Mapping and Other Applications of World Projections", *IEEE Computer Graphics & Applications*, **6**(11):21-29, 1986.
7. P.S. Heckbert, "Survey of Texture Mapping", *IEEE Computer Graphics & Applications*, **6**(11):56-67, 1986.
8. Y. Horry, K. Anjyo, and K. Arai, "Tour Into the Picture: Using Spidery Mesh Interface to Make Animation from a Single Image", *Proceedings of SIGGRAPH 97*, 225-232, 1997.
9. H.W. Kang, S.H. Pyo, K. Anjyo, and S.Y. Shin, "Tour Into the Picture Using a Vanishing Line and its Extension to Panoramic Images", *Computer Graphics Forum*, 20(3):132-141, 2001.
10. D. Liebowitz, A. Criminisi, and A. Zisserman, "Creating Architectural Models from Images", *Computer Graphics Forum*, **18**(3):39-50, 1999.
11. Microsoft, *DirectX 8 Programmer's Manual*.
12. B.M. Oh, M. Chen, J. Dorsey, and F. Durand, "Image-Based Modeling and Photo Editing", *Proceedings of SIGGRAPH 2001*, 433-442, 2001.
13. H.Y. Shum and L.W. He, "Rendering with Concentric Mosaics", *Proceedings of SIGGRAPH 99*, 299-306, 1999.
14. R. Szeliski and H.Y. Shum, "Creating Full Panoramic Mosaics and Environment Maps", *Proceedings of SIGGRAPH 97*, 251-258, 1997.
15. F.M. Weinhaus and R.N. Devich, "Photogrammetric Texture Mapping onto Planar Polygons", *Graphical Models and Image Processing*, **61**(2):63-83, 1999.
16. http://www.panoguide.com/

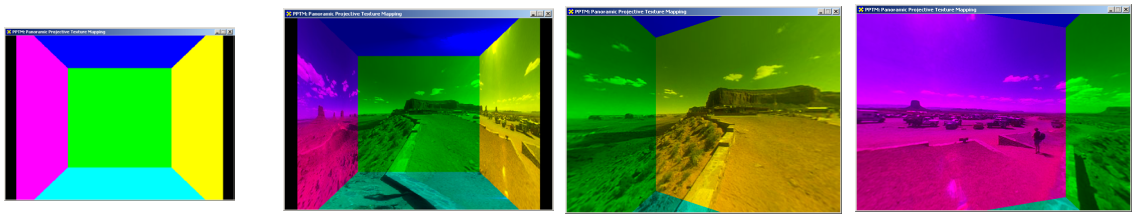
(a) (b)



(a)

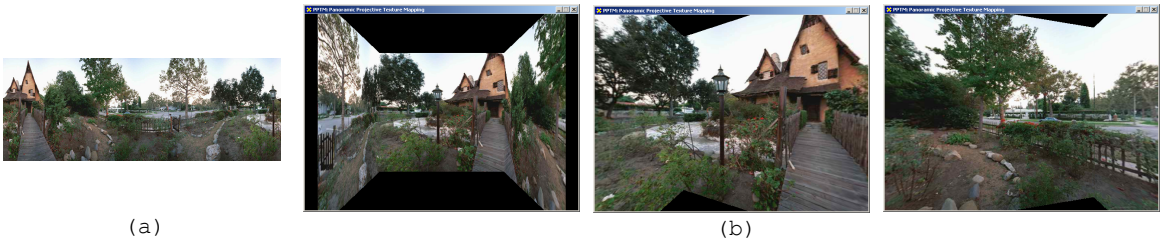
(b)

Figure 8: Result of Cylindrical Panorama Viewer



(a) (b)

Figure 9: Result of Cubic Panorama Projection



(a) (b)

Figure 10: Result of Cylindrical Panorama Projection