A GENERALIZATION APPROACH FOR 3D VIEWING DEFORMATIONS OF SINGLE-CENTER PROJECTIONS

Matthias Trapp and Jürgen Döllner
Hasso-Plattner-Institut, University of Potsdam, Germany
{matthias.trapp, juergen.doellner}@hpi.uni-potsdam.de

Keywords: Real-time Panorama, Non-Planar Projection, Fish-Eye Views, Projection Tiles.

Abstract: This paper presents a novel image-based approach to efficiently generate real-time non-planar projections of arbitrary 3D scenes such as panorama and fish-eye views. The real-time creation of such projections has a multitude of applications, e.g., in geovirtual environments and in augmented reality. Our rendering technique is based on dynamically created cube map textures in combination with shader programs that calculate the specific projections. We discuss two different approaches to create such cubemaps and introduce possible optimizations. Our technique can be applied within a single rendering pass, is easy to implement, and exploits the capability of modern programmable graphics hardware completely. Further, we present an approach to customize and combine different planar as well as non-planar projections. We have integrated our technique into an existing real-time rendering framework and demonstrate its performance on large scale datasets such as virtual 3D city and terrain models.

1 INTRODUCTION

This work introduces a concept to compensate for the field-of-view (FOV) limitations of the classical pin-hole camera rendering pipeline. It has been developed to enable the application of non-planar projection on standard consumer graphics hardware in real-time. Examples are omni-directional panorama for non-planar screens or spherical dome projections (Bourke, 2004). This work focuses on real-time modifications of perspective views that are possible due to the recent hardware developments (Blythe, 2006).

The research in the field of non-planar or non-linear projection distinguishes mainly between two different projection types: projections with a single projection center (SCOP) (Carlbom and Paciorek, 1978) or with multiple projection centers (MCOP) (Rademacher and Bishop, 1998). Our approach is limited to SCOP. In spite of non-planar projections screens (Nelson, 1983), this technique can also be used for rendering effects in games as well as to improve visibility if used in virtual landscapes (Rase, 1997) or city environments. The user can benefit from extreme perspectives (Glaeser, 1999) and large FOV angles by having a reasonable survey of the scene (Glaeser and Gröller, 1999) as well as a better size and depth perception (Polack-Wahl et al., 1997). Our method exploits the technique of dynamic environment mapping (in terms of cube map texturing) in combination with the programmable GPU. The separation of projection calculation and cube map texture creation enables a broad range of optimization techniques.

Figure 1: A cylindrical projection with a horizontal FOV of 270° and a vertical FOV of 90°.

Existing image based approaches for non-planar projections suffer mainly from the lack of interactive capabilities if used with complex geometric scenes such as virtual 3D city models or for large viewports. This can be explained by the trade-off between the generality of the proposed frameworks and their efficient reproduction. Furthermore, the parameterizations are complex and cannot be controlled by the user intuitively (Brosz et al., 2007).

Our approach was inspired by (van Oortmerssen,
2 RELATED WORK

This section gives an overview of research in the fields of SCOP and MCOP projections. There is a vast amount of literature covering foundations and applications of non-planar as well as non-linear projections. In (Brosz et al., 2007) a sophisticated overview is presented.

2.1 SCOP Approaches

To achieve distortions or special projections of the 3D scene, the pinhole camera is extended in several ways. In (Bayarri, 1995) a procedure is proposed that is based on the computation of new absolute coordinates to be transformed through an adaptive projection matrix. A flexible adaptive projection framework is described by Brosz et al. (Brosz et al., 2007) that enables the modeling the modeling of linear, non-linear, and hand-tailored artistic projections. It uses ray-casting and scanline rendering algorithm where polygonal coordinates are changed by a vertex shader. The generality of the framework makes efficient projection difficulty, especially for large scale scenes. Distortions as sub-category of geometric registration or image warping are discussed in (Glasbey and Mardia, 1989) and (Glasbey and Mardia, 1989).

2.2 MCOP Approaches

Additional to SCOP imaging there are non-planar projection surfaces that require multiple perspectives, e.g., a number of images from different center of projections (Wood et al., 1997). The goal is to keep qualities of global scene coherence, local distortions, and shading results from the changes in perspective of each projection. View-independent rendering, expressive CG imagery or animation are just a few applications for this area of scientific research.

Apart from slit cameras, (Glassner, 2000) introduced the Cubist Camera that presents many interpretations and points of view simultaneously. The technique uses nonlinear ray tracing that handles lighting but can cause artifacts. Given a scene geometry and an UI to position local and master cameras (Agrawala et al., 2000), Artistic Multiprojection Rendering provides a tool for creating multi-projection images and animations. The principal item is the algorithm that solves occlusions of the scene objects, each rendered by a local camera. This is able to handle fixed distortions and creates surrealistic, toony styles but cannot solve lighting and shadow problems.

A fresh Perspective (Singh, 2002) introduces also an interactive approach that does neither handle illumination issues nor control global scene coherence. The difference: the resulting nonlinear perspective image
of an object is potentially influenced by all cameras. Constitutive on that is RYAN (Coleman and Singh, 2004). This interactive system integrates into the conventional animation work flow. It distorts the scene geometry prior to the linear perspective transformation. The result will appear nonlinearly projected. Just like (Agrawala et al., 2000), it uses two kinds of cameras: boss (traditional linear perspective) and lackey (represent local linear views). The illumination is done by blending illumination of boss and lackey cameras or setting a single view point for lighting.

3 BASIC CONCEPT

The concept of our approach is based on two components: a dynamic environment map (Heidrich and Seidel, 1998) and fragment shader functionality (NVIDIA, 2005). First, the virtual environment is rendered into a cube map texture. The cube map texture is a fully hardware accelerated feature (NVIDIA, 2005). First, the virtual environment is rendered into a cube map texture. The cube map texture is created by transforming it with respect to the current camera parameters. The transformation matrix $C$ is an orthonormal base constructed from the current look-to vector $L_T = (x_T, y_T, z_T) \in \mathbb{R}^3$, look-up vector $L_U = (x_U, y_U, z_U) \in \mathbb{R}^3$ and the cross product $L_C = (x_C, y_C, z_C) = L_T \times L_U$ so that:

$$C = \begin{bmatrix} x_T & y_T & z_T \\ x_U & y_U & z_U \\ x_C & y_C & z_C \end{bmatrix} \iff L_T \cdot L_U \neq 0$$

Following this, the final sampling vector $V$ for a projection $\mathcal{P}$ and a fragment $F_a$ is calculated via:

$$V = (C \cdot \delta_{\mathcal{P}}(F_a \cdot s)) - O$$

The vector $O \in \mathbb{R}^3$ is denoted as off-axis vector (Bourke, 2004). Figure 2.C demonstrates an example for an off-axis projection. The scalar term $s \in \mathbb{D}$ can be interpreted as a zooming parameter. To avoid unnecessary calculations at runtime, the distortion vector $V$ can be stored into a high-precision floating-point normal map (Kilgard, 2004). This reduces the shader execution costs for projection calculation to two texture look-up operations and enables a new technique for combining different projection using projection tiles.

4 PROJECTION TILES

This method enables the combination of planar projections and different non-planar derivatives as well as...
it facilitates the creation of custom projections which are hard to describe analytically. In this context, projection tiles are a generalization of the concepts described in Section 3. They enable control over different projection types as well as add a smooth transition of their parameters. A projection tile defines the area of a specific projection in relation to the viewport. Projection tiles are organized in a projection tile screen (PTS) (Figure 4.A). It is represented by a specific feature set \( \mathcal{T}^{\text{proj}} \) defined as follows:

\[
\mathcal{T}^{\text{proj}} = \begin{bmatrix}
E_{0n} & \cdots & E_{mn} \\
\vdots & \ddots & \vdots \\
E_{00} & \cdots & E_{m0}
\end{bmatrix}
\]

A component \( E_{ij} \) of \( \mathcal{T}^{\text{proj}} \) is denoted as tile feature. It basically describes a view direction for a specific point on the screen. To enable a more intuitive way to describe a tile screen, we have chosen spherical polar coordinates \( \phi \) and \( \theta \) to express the view direction instead of a normalized direction vector. Thus, we define a feature \( E_{ij} \) as a 6-tuple:

\[
E_{ij} = (x_{ij}, y_{ij}, e_A(\theta_{ij}), e_A(\theta_{ij}), s_{ij}, f_{ij})
\]

that contains the feature position \((x_{ij}, y_{ij})\) and the particular horizontal and vertical view angles \((\theta_{ij}, \theta_{ij})\). The parameter \( s_{ij} \) is the respective scaling factor while the variable \( f_{ij} \) can be used to bind custom parameters to each feature. Figure 5 demonstrates this by adjusting the image saturation according to the respect value of \( f \). The function \( e_A(x) = (x + 360)/720 \) is used to encode an angle into a range of \([0, 1]\).

The PTS is transformed into a feature map (Figure 4.B) by rendering its respective tiles \( T_{kl} \) with \( k = 0 \ldots m - 1, \ l = 0 \ldots n - 1 \):

\[
T_{kl} = (E_{k(t)}, E_{k(t+1)}, E_{k(t+1),l}, E_{k(t+1),l+1})
\]

into a texture using render-to-texture (Wynn, 2002). The components \( x_{ij} \) and \( y_{ij} \) are interpreted as the 2D vertices of a quad in standard orthographic parallel projection (Woo et al., 1999). The angles and the scale factor are encoded into a per-vertex RGBA color value. The hyperbolic interpolation (Blinn, 1992) between these values is performed by graphics hardware.

The rendered feature map stores the interpolated horizontal and vertical angles \( A_x = (\phi_{st}, \theta_{st}) \) for each fragment \( F_{st} \). The normal \( N_{st} \) can be calculated by:

\[
N_{st} = f_{\text{normal}}(A_{st}) = R_z(s^{-1}_A(\theta_{st}))(R_z(e^{-1}_A(\phi_{st}))[0][0] + s \ t \ 0)
\]

Where \( R_z \) and \( R_s \) denote the 3D rotation around the respective axis. For each \( N_{st} \) in the resulting normal map (Figure 4.C) a sampling vector \( S \) can be calculated by setting \( \delta_{\text{proj}}(F_{st} : s_{st}) = N_{st} \). The result is shown in Figure 4.D.

5 IMPLEMENTATION

The rendering of non-planar projections is performed in two steps per frame:

1. Create or update dynamic cube map as described in Section 5.1. During this phase, optimization methods as described in Section 5.2 can be utilized.

2. Apply projections by exploiting programmable graphics hardware (see Section 5.3). This step is applied in postprocessing and requires an additional rendering of the scene geometry.

The exemplary implementation was done by using OpenGL (Segal and Akeley, 2004) in combination with GLSL (John Kessenich, 2004). It uses framebuffer objects, floating point textures, and mip-mapped cube maps for dynamic texturing (Harris, 2004).
5.1 Cube Map Creation

An advantage of our image-based concept is the application of dynamic cube map textures (Greene, 1986). The creation of those can be optimized depending on the respective projections. This can be achieved using two techniques:

**Single-Pass Creation:** On current hardware, it is possible to create cube map textures within a single pass by utilizing geometry shaders (Microsoft, 2007). This render-to-cube-map technique duplicates each input triangle six times and applies a separate model-view transformation for each face of the cube. Each of the resulting triangles is directed to the appropriate element of a render target array view of the cube map (Blythe, 2006).

**Multi-Pass Creation:** One can create a cube map texture using multi-pass rendering in combination with render-to-texture (Gödeke, 2005). Given a reference camera position, we can construct six local virtual cameras with a FOV of 90 degrees, an aspect ratio of 1, and render the scene into the respective cube map texture targets.

We have two alternatives to construct these virtual local cameras: 1) by rotating the reference camera or 2) by using fixed cube map orientation (Figure 3). The latter demands for a look-to correction as described in Section 3 and is necessary for a simple implementation of projection tiles.

5.2 Optimization Techniques

Depending on the scene complexity, the creation of a cube map texture can become costly. Usually, the creation is fill-limited due to the increased rasterization effort as well as bandwidth-limited due to the number of rendering passes. In our application, it is not possible to use proxy geometry to speed up the creation process or distribute the rendering of each cube map face to different frames. We have implemented two main optimization techniques that are able to compensate this problem.

**CMF Optimization:** This optimization omits the update of cube map faces (CMF) that are not visible in the generated projection. To determine which cube map faces have to be updated, we calculate 360 degree spherical coordinates for each corner vertex of the unit cube and the current look-to vector \( L_T \). Then, we define a non-planar view frustum (NVF) by offsetting the spherical coordinates of \( L_T \) with the horizontal and vertical angle of the current projection. If one of the face vertices is inside the projection region, the associated face will be updated. Figure 6 demonstrates this by considering the lower vertices of the unit cube only. For the current look-to vector \( L_T \) the update of CMF \(_{\text{back}}\) is omitted.

Figure 6: Parameter space for CMF-Optimization. The cube map is displayed from the positive y-axis. The gray area represents the non-planar view frustum of a non-planar projection.
Look-To Optimization: Another optimization method would omit the update of the cube map texture if the camera position is not changed. A further optimization method can orient the cube map in such way that an optimal coverage for the respective projection region can be achieved. This includes changes in the projection function and is left for future work.

5.3 Applying Projections

The projection is rendered in a post-processing pass subsequent to the cube map creation pass(es). It can be performed according to the following steps:

1. Setup a standard 2D orthographic parallel projection with \( \text{left} = \text{bottom} = 0 \) and \( \text{right} = \text{top} = 1 \). The camera is set to the standard orientation with a look-to vector \( \mathbf{L}_T = (0, 0, -1) \).
2. Activate a specific fragment program that implements the mathematical concepts as described in Sections 3 and 4. Then, the shader program performs cube map texture lookups or outputs the calculated normal vectors for later re-use.
3. Render a screen aligned quad with standard texture coordinates that covers the entire screen. To apply the described concepts, the fragment position \((s, t)\) must be normalized in \( \mathbb{D} \).

6 RESULTS

Figures 1 and 2 show the application of our approach for the interactive visualization of large scale datasets such as 3D virtual city models. Currently, we did not apply our method to scenes made of real-world images. However, this is possible because our technique is independent from the scene representation. Figure 7 demonstrates the combination of our rendering technique together with color post-processing filters. Figure 9 shows the application to projection systems. Here, the pincushion distortion is compensated by the projectors.

Performance: Figure 8 shows the measured frame rate for 3 models of different polygon counts. The measurements were taken on NVIDIA GeForce 8800 GTX GPU with 786MB RAM and Athlon\(^{TM}\) 62 X2 Dual Core 4200+ with 2.21 GHz and 2 GB of main memory at a viewport resolution of 1600x1200 pixel. It uses multi-pass rendering to create the cube map texture. The test application does not utilize the second CPU core. The most complex sample dataset (Model 1) comprises the inner city of Berlin with about 16,000 generically textured buildings, about 100 landmarks, and a 3 GB color aerial photo on top of a digital terrain model.

Limitations: The observable subside of the frame between the cube map resolution 512 and 1024 pixel is caused by a fill limitation of the used graphics hardware and can vary. The necessary resolution of the cube map texture depends on the resolution of the viewport and the used projection. For high horizontal and vertical FOV, a cube map resolution of 1024\(^2\) pixels is sufficient in most cases.

The quality of the feature map depends on the resolution of the PTS. If the resolution is too low, the feature interpolation can cause artifacts for tiles with acute angles. We obtained blurred output images for extreme FOV. This is the key disadvantage of our approach and is caused by texture sampling artefacts.
7 \textbf{FUTURE WORK}

Although our approach performs as expected, there are still possibilities for further improvements. The adaption of cube map optimization techniques to the single pass creation technique as well as experiments with irregular texture resolutions are left for future work. To compensate for unsharp output images we are up to develop custom filtering methods for cube map textures.

We are particularly interested in supporting other SCOP projections as described in (Margaret, 1995). The representation of standard non-planar projections using tile screens enables the combination of non-planar projections with the 2D lens metaphor as described in (Spindler et al., 2006; Carpendale and Montagnese, 2001) as well as (Bier et al., 1993). Therefore, a authoring tool for PTS is necessary. Further, we try to adapt our rendering technique to generate non-planar for 3D anaglyph images.

Additionally, we can apply the isocube approach (Wan et al., 2007) to compensate cube map sampling artifacts for lower resolutions. Finally, a comparative performance evaluation concerning time efficiency of our concept compared to other approaches is left for future work.

8 \textbf{CONCLUSIONS}

We have presented an approach to generate non-planar projections for real-time applications. We suggested optimization methods to accelerate the creation of dynamic environment maps and introduced a novel approach for creating combined SCOP projections. We demonstrated the results and performance of our rendering technique by integrating it into an existing rendering framework (3Dgeo, 2007).

\section*{ACKNOWLEDGEMENTS}

This work has been funded by the German Federal Ministry of Education and Research (BMBF) as part of the InnoProfile research group ’3D Geoinformation’ (http://www.3dgi.de). The authors would like to thank also 3D geo for providing the texture data as well as the cadastral data of virtual 3D city models.

\section*{REFERENCES}


