UNIVERSITY OF CALGARY

The Flexible Projection Framework

by

John David Lynn Brosz

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF COMPUTER SCIENCE

CALGARY, ALBERTA

April, 2011

Abstract

Three dimensional computer graphics are used extensively in visual media. A key reason for this use is the computer's ability to quickly and accurately project 3D environments onto 2D images. Understanding of this process is crucial to organizing 3D space in a 2D image. This thesis contributes to this understanding by exploring the representation of a wide variety of projections within a new framework, Flexible Projection (FP).

Unfortunately the standard graphics pipeline only supports linear projection; this leaves a wide variety of nonlinear projections (e.g., fisheye, panoramas) unavailable to most image creators. FP not only provides a means for including nonlinear projections in graphics systems, but also provides a consistent means of creating and working with a wide variety of nonlinear projections rather than providing a single, one-off solution.

The approach taken with FP is to explicitly model the projection's viewing volume parametrically. Beginning with a cube one can flexibly adjust this volume into other shapes: a frustum for perspective, a cylinder for a cylindrical panorama, a hemisphere for a fisheye. This allows for a diverse variety of projections with a relation between the volume's shape and the resulting image properties. This creates a link that can assist in communicating the projection's behaviour without resorting to mathematical equations.

Two approaches for rendering these viewing volumes are described: ray-casting and a less accurate approach that makes use of scanline rendering for real-time results. To demonstrate the utility of FP we describe how it can be used to reproduce a wide variety of projections from current literature as well as in creating new projections. Additionally, we demonstrate several applications: cameras that animate transitions between different projections, modeling a projection from a painting and applying this projection to other scenes, as well as adapting the framework for the express purpose of creating customizable panoramas.

Acknowledgements

There are many people to whom thanks are due for their assistance in this research.

I would like to first thank my supervisor, Faramarz Samavati for his ongoing support, collaboration, teaching, and his close examination of my math. I am ever-so-grateful for his efforts in refining my work that have made the defense a C^∞ smooth experience.

A great deal of thanks are also due to Sheelagh Carpendale for her ongoing support of this research through collaboration and funding but, also importantly, her excitement over and her insights into this topic. I very appreciative of her small push that moved me to change my thesis topic to this area.

I am also grateful to the various members of my candidacy and supervision defense committees: Mario Costa Sousa for finding me to a wide variety of related work, Jon Rokne for his appreciation of the mathematical, Richard Levy for pointing me towards a variety of applications, and my external examiner Pierre Boulanger for his enthusiasm and ideas for expanding the scope.

Throughout the last six years at the University of Calgary I greatly enjoyed and benefited from contact with my fellow students and associates: Joseph Cherlin (who initiated this project by wondering what an image created with curved light would look like), Luke Olsen, Miguel Nacenta, Erwin de Groot, Adam Runions, Katayoon Etemad, Lindsay MacDonald, Hao Wang, Richard Pusch, and Javad Sadeghi. Thanks go out to the members of the Jungle and Innovis Labs as a whole.

Extra thanks are due to Luke Olsen, Ruth Hart Budd, Kristy Brosz, Helen Davidson, and Kristian Wasén for their assistance in proof reading.

Financial support for my studies and this research has been provided by Natural Sciences and Engineering Research Council of Canada (NSERC), iCore, UTI Technolo-

gies, and the Department of Computer Science, University of Calgary and is gratefully acknowledged.

Finally, this thesis owes a great deal to the support of my family. Kristy has put up with many late nights, tirades of research talk, and has been kind enough to firmly inserted my nose back onto the grindstone as needed. She takes great care of me. Thanks also to Kristy's parents Wayne and Jeanne Burnham for their support. Lastly and most importantly, eternal thanks are due to my parents, Liz and Roy Davidson, for everything.

Table of Contents

List of Figures

Chapter 1

Introduction

Three dimensional (3D) computer graphics are used extensively in visual media. One of the key reasons that computer graphics have found such broad support in visual media is that computers can quickly and accurately create flat 2D images from complicated 3D environments. This is done via an operation known as projection. The accuracy of projection, in comparison to hand-drawn images, is important because, when projection is consistently performed over consecutive images, it provides the illusion of a three dimensional environment with predictable depth and scale. The Disney movie *Beauty and the Beast* is one of the first well-known examples where computer graphics were employed to merge hand-drawn animated characters with a synthetic 3D environment. The ballroom scene in this movie (Figure 1.1) in particular illustrates the use of accurate projection to maintain the illusion of a three dimensional space.

Figure 1.1: Screenshot from *Beauty and the Beast* (2002, Special Edition DVD). Film ©1991, 2002 The Walt Disney Company.

The concept of projection is not limited to computer graphics. Generally speaking, projection is the process of transforming from one space to another, usually from a higher dimension to a lower dimension. This thesis is primarily concerned with projection from three dimensions to two dimensions, for the purpose of creating images from three dimen-

sional environments. There are many different possibilities when choosing a projection, a choice that can greatly affect the impression of the resulting image. Some projections can make images seem very realistic, while others can create a surrealistic impression. Some images make viewers feel like they are within or a part of the image, other clearly express a divide between the viewer, and the scene being portrayed.

The scientific approach to projection is to view it as the mathematical reduction of dimension coupled with an enforced organization of the reduced dimension's space. This leads to derived geometries, spaces, and formulae used to control this operation. Conversely the artistic view of projection treats this operation as a means of organizing image elements, relationships between objects, and the observer's presence relative to the image.

When considering the use of projection in computer graphics, it is critical to note that only linear projections are generally supported. That is, the usual scanline-based rendering techniques used in computer graphics require that projections be represented as matrices that operate upon homogeneous coordinates [Gol02]. This allows projection to be treated similarly to other transformations such as rotation and translation. A second and more critical requirement is that these projections map straight lines to either lines or points, but not curves. This allows for the use of linear interpolation for clipping, filling, and texturing operations that would be considerably less efficient if they had to handle curved polygon edges.

These limitations effectively cause conventional scanline rendering algorithms to be limited to perspective and parallel projections. While these are powerful projections with a variety of uses, they impose severe restrictions on the role that projection can assume in the process of image creation.

Projections that do not meet these two requirements are generally known as nonlinear projections. Commonly known nonlinear projections include panoramic, fisheye, and

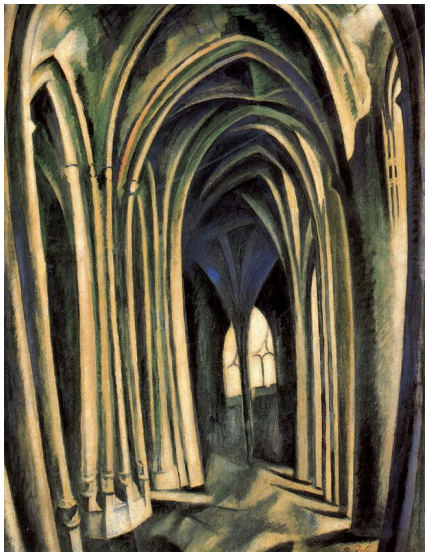


Figure 1.2: *Saint-Séverin No. 3*, Robert Delaunay, 1910.

Figure 1.3: A portion of the *Ghent Altarpiece*, Jan van Eyck, 1432.

map projections. Such projections allow interesting and useful views of 3D geometry. For example, both fisheye and panoramic projections allow us to expand a view of our surroundings from the small limited window of perspective, to a far larger field of view. Map projections handle the difficult task of stretching and distorting the surface of a sphere so that it can be displayed upon a planar surface.

Another argument for supporting the use of nonlinear projections is that even though artists have had a strong formulation of perspective since the Renaissance, they have made use of nonlinear projections prolifically despite the many strengths of perspective. Extreme examples of nonlinear projections in Western art include cubist and expressionistic forms of modern art. Figure 1.2 provides a strong example of artistic use of nonlinear projection. Artists make use of different projections deliberately for specific effects and purposes. For instance, Hockney notes that the projection used in *The Ghent Altarpiece* (Fig. 1.3) creates a greater feeling of closeness to the scene in its entirety than a perspective projection would [Hoc06, p. 94].

Anyone who has used a camera is aware of the difference between the image captured

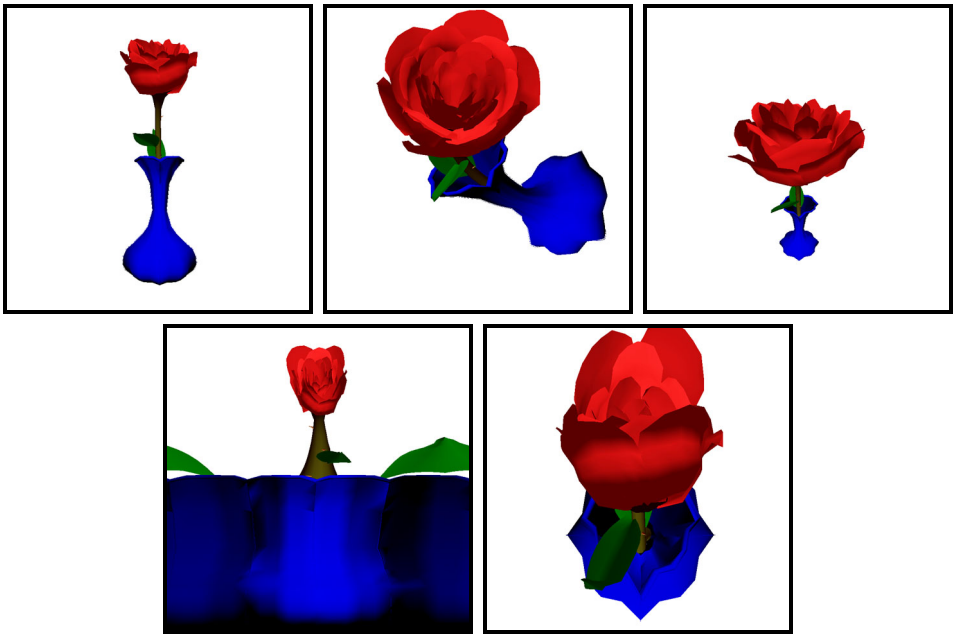


Figure 1.4: Five different projections of a rose.

by a camera and their own perception of the same scene. The aim of many nonlinear projections is to bridge this difference – to create projections that emphasize particular aspects of the three dimensional scene that the image creator perceives as important. For instance, consider the different projections of a rose in Figure 1.4: each projected image provides a different understanding of the rose. In this way projection provides control for image creators to create images that reflect their perception. Control over projection reveals the differences between perception and image creation; perceived reality and perspective.

1.1 Goals

The preceding section has established that projection is a crucial operation in computer graphics, yet the rigidity of the standard rendering pipeline makes it difficult to use nonlinear projections. Now we can begin to examine the question of how nonlinear projections can be used within computer graphics.

In order to do so we must first look at how nonlinear projections are defined. When painting or drawing, artists have explicit control over all aspects of image creation. Consequently they can create nonlinear projections without fixed rules, but rather are based on what they believe to be the most effective projection for their purposes. The downside to this approach is that it requires strong technical skills, is hard to remain accurate, and is time consuming to modify and experiment with different projections. Additionally it is extremely difficult – if not impossible – for others to reproduce an interesting nonlinear projection in other scenes and images. In general, these artist-created projections cannot be bound to fixed matrices or even nonlinear equations. Bringing these possibilities for fluid expression through projection into computer graphics requires a different way to define projections.

Mathematically derived projections such as a cylindrical panorama are usually defined in terms of a projection equation that, for given parameters (e.g., the cylinder's dimensions and position) transforms 3D coordinates into 2D coordinates. Such equations provide an accurate projection that can be uniformly applied to an entire scene. While these derived projection equations are useful once derived, they make experimentation outside the parameters of the equation exceedingly difficult. This results in the image creator settling for a less than perfect projection, since developing new projection equations is time-consuming and difficult.

It is clear that neither of these approaches provides an environment suitable for mastery of nonlinear projections. The artist approach is very low level, is applied informally, and does not take advantage of the calculation power that computers offer. Projection equations are essentially "one-off" solutions that are rigid within their parameters and cannot be combined or merged with other types of projection. This leads us to the first goal of a nonlinear projection framework: to provide some means of describing

projections that lies between the "creating the entire image from scratch" and various projection equations.

In computer graphics literature there exists a variety of descriptions for individual nonlinear projections, each suited for a specific purpose as will be outlined in Chapter 3. These individual projections are generally defined through specific projection equations and exist as point solutions in the entire space of possible nonlinear projections. The drawback of these individual solutions is that they can only be considered independently of one another and each has its own implementation requirements. This leads us to another goal: a nonlinear projection system should encompass many of these nonlinear projections to enable exploration in scenarios when one is not sure which nonlinear projection best suits their goals and requirements.

A further limitation of existing techniques is that each has its own approach to integration with the computer graphics pipeline, such as requiring ray-tracing [Gla00] or blending together images from several perspective projections [AZM00]. Therefore another motivation for this work is through bringing all nonlinear projections into a single framework, rendering techniques developed for the framework as a whole, can then be applied to a large number of individual nonlinear projections in a uniform fashion. It follows then, for this research, to describe a rendering technique that can be applied to these projections.

To summarize, the goals for this research include:

- Developing an approach to defining projections that, like projections created by artists, allows more flexibility as to how projection is performed while maintaining the accuracy and repeatability of mathematically derived projections;

- Creating a single framework capable of representing a large set of all possible projections;

- Developing techniques, both for describing projections as well as for rendering them, that can be applied to many different projections and executed with computer graphics hardware.

1.2 Contributions & Methodology

In order to meet these goals we introduce a framework for designing linear and nonlinear projections that allows a wide variety of nonlinear projections to be more easily utilized in computer graphics. This framework introduces a unified geometry for linear, nonlinear, and hand-tailored artistic projections through the concept of a flexible viewing volume that is modeled with a parametric representation (see Figure 1.5). For this reason we have named this framework Flexible Projection. With this representation we move away from the standard frustum of a perspective projection to allow for projections without a specific eye position, curved projection surfaces, and even volumes that curve throughout.

This parametric approach to modeling the viewing volume immediately introduces two benefits: the first is that it allows Flexible Projection to make use of a wide variety of already-existing modeling tools for designing the viewing volumes instead of deriving projection equations. Second, it makes it possible to visualize these projections, providing information about how a given projection works based on its appearance.

The Flexible Projection Framework's definition is quite general, providing a wide variety of opportunities to define projection volumes in different ways. In order to make use of the framework, we describe two different approaches to interfaces for defining projections volumes that we have found useful in practice. While these interfaces are by no means the only possible mechanisms for defining viewing volumes, they do provide a useful starting point for exploring the possibilities of Flexible Projection.

To demonstrate the usefulness of Flexible Projection we explore several different ap-

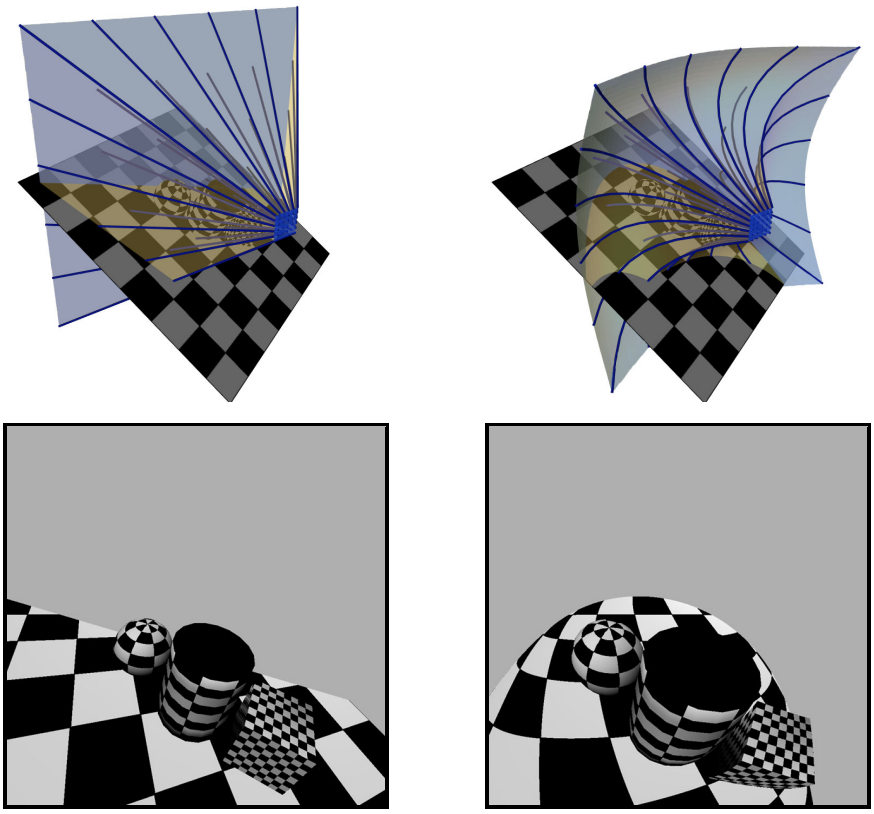


Figure 1.5: Two parametrically defined projection volumes (top) and the resulting projected images (bottom). The left projection is a perspective projection, the right is a nonlinear projection.

plications. One such example is the ability to recreate projections used by artists, a difficult task that is not feasible without an interactive system. Another is the treatment of viewing volumes as objects that can easily be animated, key-framed, and interpolated to create a variety of new camera movements and effects. An in-depth investigation into an interface for creating and rendering panoramas is also described.

The contributions also include two different techniques for rendering Flexible Projections. The simplest approach is based on ray casting, where rays are generated by isolating iso-curves from the parametric viewing volume. This requires the standard ray-casting approach to be modified to handle curved rays. In particular we introduce a new type of intersection test, between quadratic curves and triangles that can yield significant time savings when rendering nonlinear rays. The second technique makes use of scan-

line rendering, allowing integration of nonlinear projections into the standard graphics pipeline. This is accomplished through the use of programmable shaders in graphics hardware and, while this technique is limited to specific types of projections, it provides realtime rendering for complex models and projections.

Lastly, this thesis also provides two contributions related to general understanding of linear and nonlinear projections. First it describes an explicit test for differentiating between linear projections (i.e., those projections that work within the standard computer graphics rendering pipeline) and nonlinear projection (i.e., those that do not). Second, detailed comparisons are made between existing nonlinear projection techniques and the Flexible Projection Framework. Details are also provided on how these existing techniques can be reproduced as Flexible Projections.

1.3 Thesis Organization

This thesis is organized into ten chapters. Chapter 2 closely examines the differences between linear and nonlinear projections; providing a test for distinguishing the two. Chapter 3 discusses background material related to projection and more specifically nonlinear projection. Chapter 4 introduces and describes the Flexible Projection Framework while Chapter 5 describes two interfaces for defining Flexible Projections. In Chapter 6 we describe how a wide variety of nonlinear projections can be reproduced with Flexible Projection. Additionally we examine, compare, and contrast the differences between Flexible Projection and a category of projections created by blending the output of several projections together. Chapter 7 describes how to render Flexible Projections ray casting and through the current graphics pipeline with GPU programming. Chapter 8 outlines several applications of Flexible Projections that have been investigated over the course of this work. Chapter 9 closely examines one particular area of application, pro-

viding interface and rendering techniques for creating novel and customizable panoramic projections. Lastly, Chapter 10 summarizes the thesis' contributions and discusses future avenues of research.

Chapter 2

Differentiating Linear and Nonlinear Projections

While, as will be shown in later chapters, Flexible Projection is capable of representing both linear and nonlinear projections, it is important to differentiate between these types of projections for two reasons. The first is that the standard graphics approach to projections is designed for linear projections, thus aspects of the Flexible Projection Framework that merely reproduce these well-established projections cannot be seen as new contributions. The more important reason is that only by defining what exactly a linear projection encompasses can we differentiate what is meant by the negatively defined term[1], nonlinear projection.

This chapter also defines a variety of projection-related terms, such as the difference between a nonlinear projection and a nonlinear projector, a distinction that can greatly affect understanding of these concepts.

The chapter is organized as follows. Subsection 2.1 discusses the vocabulary that is used to describe projections. Subsection 2.2 introduces necessary concepts for distinguishing between linear and nonlinear projections. These concepts are then demonstrated for six different types of projection, four that are linear projections and two that are not.

2.1 Projection Vocabulary

In order to clearly describe projections it is useful to introduce common terms used in computer graphics when discussing projections.

The definition of *projection* that this thesis is concerned with is defined by the Ox-

[1]A negatively defined term is a term that is defined by what it is not, rather than what it is. Another computer graphics example of a negative definition is non-photorealistic rendering.

ford English dictionary as "... a representation of a figure on a surface according to a particular system of correspondence between its points and the points of the surface [or] an analogous operation performed in a space of different dimension" [OED11]. While this thesis is primarily concerned with projections that reduce three dimensional space and objects to two dimensions, any transformation that reduces the dimensionality of a space is considered a projection.

The *viewing volume* is the entire volume of the 3D scene that, if not obscured, would be projected onto the 2D image. An example is shown in Figure 2.1. This volume is bounded on the sides, usually in some relation to the dimensions of the output image. The front of the volume is either defined by the camera's position or by a near clipping surface. This *near surface* is used in computer graphics applications to prevent objects from appearing too close to the camera, thereby obscuring the majority of the scene for perspective projections. This near surface is also referred to as the projection plane or projection surface. In computer graphics the back of the volume is defined by a far clipping surface. This *far surface* allows culling of objects that are too distant from the camera to add significant impact to the projected image.

Projectors are usually lines (or sometimes curves) that pass through all the points of the volume that are projected a single point on the projection plane/image [CP78]. Some projections feature a single point where all projectors intersect one another. This point of intersection can sometimes occur outside of the viewing volume. If such a point exists it is referred to as the eye or camera position. Projections with an eye position are said to portray a single viewpoint as all projectors feature a common origin. Projections without such a point incorporate many viewing positions as each projector possesses its own origin. The *center of projection* refers to the projector at the center of the image (particularly in perspective projection).

A *projection equation* is a function that performs a projection by transforming 3D

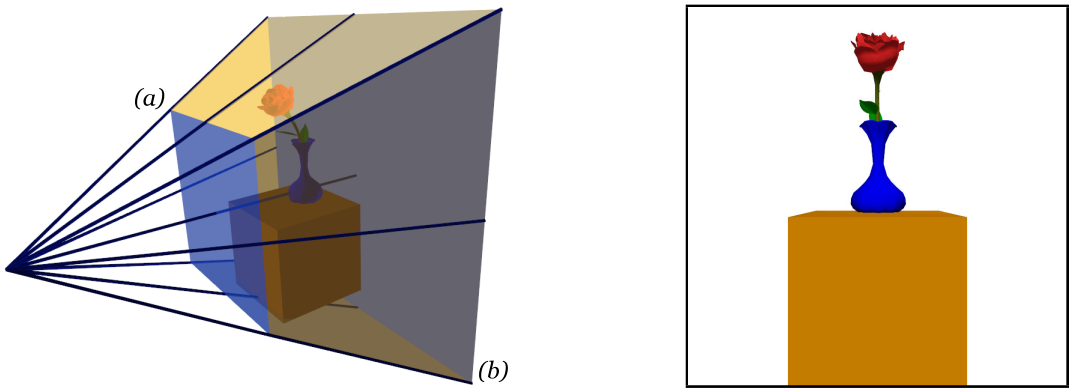


Figure 2.1: Left: The viewing volume of a perspective projection containing a scene with a rose on top of a cube. The sides of the volume are colored yellow, while the near and far surfaces of the volume are blue and have been labeled (a) and (b) respectively. A selection of projectors are shown as blue lines and have been extended in front of the near surface to illustrate the eye position. Right: the projected image.

coordinates (x, y, z) into image coordinates (x^*, y^*) and sometimes depth z^*. While the purpose of projection is to produce an image, some measure of depth is useful for other operations such as occlusion/visibility testing and interpolating vertex attributes (such as texture coordinates). These equations often assume that (x, y, z) have been rotated and translated into camera coordinates where the origin corresponds with the eye position (if present) and the z-axis is aligned with the depth of projection volume. Instead of directly producing image coordinates (x^*, y^*), projection equations often map to normalized devices coordinates (u, v, t) where all points in the viewing volume are in $\{-1 \leq u \leq 1,\ -1 \leq v \leq 1,\ 0 \leq t \leq 1\}$[1]. A separate transformation, the viewport transformation, is then used to map normalized device coordinates to screen pixel coordinates. A *map* is "a way of associated unique objects to every element in a given set. So a map $f : A \to B$ from A to B is a function f such that for every $a \in A$, there is a unique object $f(a) \in B$." [Wei11b].

[1]Note that some computer graphics systems assume different ranges for device coordinates. For instance OpenGL assumes a cube (i.e., $-1 \leq z \leq 1$) [Shr06] rather than the hemicube used above. Alternatively many print based systems use normalized device coordinates that require all axes to be in the range of 0 to 1 [Pri].

A projection's *viewing angle* refers to the difference in angle between the projectors corresponding to the extreme right and left of the image.

Last of all, the *scene* refers to the 3D geometry that is being projected into 2D.

2.2 Defining Linear Projection

Before exploring the literature on nonlinear projections it is important to differentiate between linear and nonlinear projections. This is especially important as the terms linear and nonlinear are used in conjunction with several different terms through the related literature. Examples include linear and nonlinear transformations, equations, ray tracing, projection, and projectors.

The reason this differentiation is necessary is that existing literature on this subject does not explicitly describe these definitions or, when they do, they provide different examples that may not align with other definitions. For example, Salomon in his book *Transformations and Projections in Computer Graphics* states that linear projections are *all* based on objects projected onto planes with the scene on one side of the plane, and the eye position on the other. When the eye positions is at infinity the result is a parallel projection, otherwise it is a perspective projection [Sal06, p. 3]. Later in the text Salomon states that a nonlinear projection is any projection that "cannot be expressed by linear transformations such as $x^* = ax + cy + m$ and $y^* = bx + dy + n$" [Sal06, p. 145] where $a, c, b, d, m, n \in \Re$. Note that this linear transformation requirement does not require the projection to be a 3D linear transformation (something that perspective, for instance, is not).

These definitions of linear and nonlinear projections seem straightforward but let us quickly examine an example that shows otherwise. First, begin with perspective projection, a well known linear projection. The projection equation for perspective $f(x, y, z) =$

$(\frac{x}{z}, \frac{y}{z})$. If we consider Salomon's nonlinear projection definition, this equation can be considered a linear transformations by setting $a = \frac{1}{z}, c = 0, m = 0, b = 0, d = \frac{1}{z}, n = 0$.

Now consider a pushbroom projection [GH97]. Pushbroom projection will be described in more detail in Section 2.2.5 but at this point it is only necessary to consider a simple pushbroom projection equation: $f(x, y, z) = (x, \frac{y}{z})$. According to the rules by which perspective was shown to be linear it seems that Pushbroom projection is also linear as we can set $a = 1$ and the remaining constants the same as perspective. However, if we examine pushbroom projection geometrically according to Salomon's rule for linear projection we find that a pushbroom projection does not have an eye position, rather the projectors meet at a line (for our simple pushbroom projection equation this line is at $y = 0, z = 0$). Is this projection linear or not? In Section 2.2.5 we will show that pushbroom projection should be considered a nonlinear projection. This sort of incongruity causes misunderstandings regarding what is and is not a linear projection and the purpose of this section is to make the distinction clear.

Before deciding what exactly a linear projection is, let us examine what properties of projections are important for their use in the standard graphics pipeline. There are three such properties: (1) the projection can be represented as a matrix in a homogeneous coordinate system; (2) the projection preserves collinearity; and (3) the projection maintains the ratio of distances.

In the rendering pipeline, projections as well as other transformations are implemented through 4×4 matrices and thus as linear transformations within homogeneous coordinates. Homogeneous coordinates add an additional coordinate to Cartesian coordinates, increasing dimensionality so that translation can be implemented as a matrix multiplication rather than as an addition of points and vectors [Wat00]. In homogeneous coordinates a point (x, y, z) is represented as (X, Y, Z, w) for any $w \neq 0$; the corresponding Cartesian coordinate is $(\frac{X}{w}, \frac{Y}{w}, \frac{Z}{w})$. In order to represent a projection with a

4×4 matrix, the projections equation must be in the form $x^* = ax + by + cz + dw$ and $y^* = ex + dy + fz + gw$ where $a, b, c, d, e, f, g \in \Re$ and provide the top 4×2 entries of the matrix. The remaining half of the matrix is used to control depth after projection (necessary for occlusion testing, etc) and the homogeneous coordinate. Representation in this matrix form allows projections to be calculated efficiently and be combined with other other transformations. It is important that projection be a linear transformation in 4D (unlike Salomon's previously mentioned definition where only 2D is considered) as there exist cases like the pushbroom camera where the 2D result is a linear transformation but its 3D or 4D representation is not.

Projections that preserve collinearity are those that transform collinear points to a line in the output 2D image. That is, lines in world space are mapped to lines (or sometimes a point) in image space. This is important in computer graphics as collinearity allows the use of linear interpolation to greatly speed up clipping, filling, lighting, and texturing operations while performing scanline rendering of triangles.

A projection that maintains the ratio of distances is a projection that, when applied to some point within a line segment, produces the same ratio of distances between the line segment's end points both before and after these points have been projected. This property is necessary for operations that interpolate across triangles such as texture mapping, colour interpolation, and Phong shading calculations.

Based on these three properties it seems clear that any projection that meets all of them should be considered a linear projection. Any such linear projection will integrate seamlessly with the standard graphics rendering pipeline. However, it is important to notice that these three criteria are not all of equal importance. For instance, if we had a projection that preserved collinearity and ratios of distance but was not a linear transformation in homogeneous coordinates (i.e., representable as a 4×4 matrix) it would be still relatively easy to adapt the graphics pipeline to accommodate such a projec-

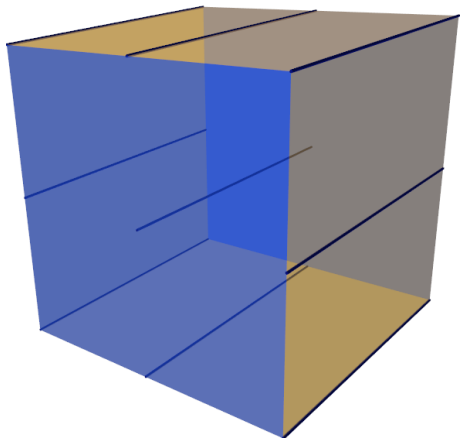


Figure 2.2: An orthographic projection volume.

tion. For instance, a programmable shader could, at the cost of some efficiency, ignore the projection matrix and implement the nonlinear projection. However, attempting to compensate for a non-collinear projection requires much more effort, requiring changes to clipping, filling, lighting, and texturing algorithms.

Based on these criteria one evident candidate of a linear projection is a projection that is an affine transformation since, by definition, affine transformations are linear transformations (in homogeneous coordinates) that preserve collinearity and the ratios of distances [Wei11a]. However, as will be shown with perspective projection, non-affine projections can also be classified as linear projections.

Now that we have identified criteria for identifying linear projections we will examine how they apply to several different projections, beginning with orthographic and perspective projection, investigating two problematic projections: inverse perspective and pushbroom, and finishing with the clearly nonlinear cylindrical projection.

2.2.1 Orthographic Projection

Orthographic projections as well as oblique projections (Section 2.2.2) are known as parallel projections. Parallel projections include all projections where the near and far

surfaces are planes and the projectors are both equally spaced and parallel to one another. Parallel projections are often used in engineering and architectural figures as well as drawings such as blueprints and schematics because these projections maintain metric properties such as distance and scale [CP78].

Orthographic projections are parallel projections that maintain a perpendicular angle between the near and far planes and the projectors' direction. The simplest of these is obtained by merely discarding the depth: $(x^*, y^*) = (x, y)$. When the projection planes are aligned with the primary axis of the scene, this projection is called orthographic; when not aligned with the scene's primary axis this projection is known as axonometric [CP78]. This distinction based on the orientation of the scene, especially if the scene has a rectangular shape, is important because angles and distances can be measured exactly when flat faces are parallel to the projection planes. Axonometric projections are useful in portraying a three dimensional overview of an object [CP78].

The viewing volume of an orthographic projection is a geometric prism where the far surface is the near surface offset along the z-axis, as shown in Figure 2.2. The projection equation for this is:

$$f(x, y, z) = (x, y).$$

Note that this is equation describes the absolute simplest scenario, in practical implementation this equation requires additional complexity for scaling to map the desired volume to the appropriate normalized device coordinates as well as translation to center the projection. This simplification will not have impact on our analysis of criteria for linear projection since scaling, rotation, and translation are all affine transformations.

Depth can be mapped from the range $[n, f]$ to $[0, 1]$ with

$$z^* = \frac{z - n}{f - n}$$

where n and f are the respective distances to the near and far surfaces and it is assumed

that $n < f$. The corresponding matrix is:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{f-n} & \frac{-n}{f-n} \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Through examination of the this matrix it is clear that it presents an affine transformation since it is the concatenation two affine transformations: a non-uniform scale and a translation along the z-axis. Consequently it is clear that lines and their ratios of distances are preserved.

2.2.2 Oblique Projection

Oblique projections are parallel projections that do not feature a perpendicular intersection between the projection's planes and projectors. The viewing volume is similar to that of the orthographic projection, but in addition to being offset by depth the far surface is also translated in the x or y directions. Figure 2.3 shows an oblique projection that features a displacement in y.

Unlike orthographic or axonometric projections, oblique projections are suited to objects with cylindrical shapes, interiors and exterior overviews of buildings, and for depicting buildings in maps [CP78]. An example of an oblique projection used for a map is shown in Figure 2.4. Carlbom and Placiorek note that proper use of oblique projections must consider the orientation of objects in the scene. Consequently they relate the rules for positioning the "projection plane so that (1) it is parallel to the most irregular of the principal faces or to the one which contains circular or curved surfaces; or (2) it is parallel to the long principal face of the object" [CP78, p.484]. Examples illustrating these rules are shown in Figure 2.5.

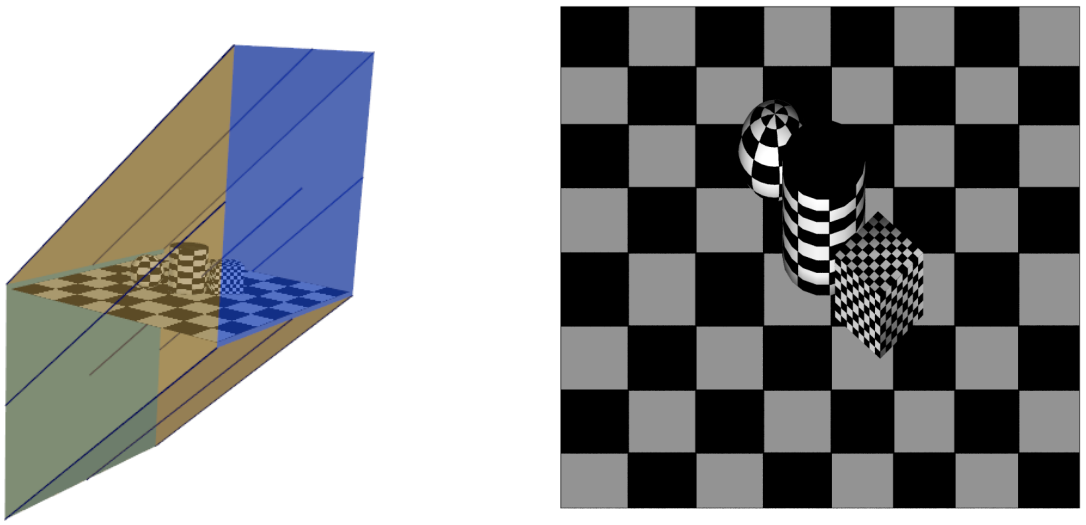


Figure 2.3: Left: an oblique viewing volume with an embedded scene (image created with a perspective projection). Right: the resulting projected image.

An example oblique projection equation is

$$f(x, y, z) = (x + \alpha z, y + \beta z).$$

Where α and β are scalars proportional to displacement of the far surfaces. The corresponding matrix is:

$$\begin{bmatrix} 1 & 0 & \alpha & 0 \\ 0 & 1 & \beta & 0 \\ 0 & 0 & \frac{1}{f-n} & \frac{-n}{f-n} \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

In examining this matrix as well as the shape of the volume in Figure 2.3 indicates the oblique projection merely incorporates a shear in what would otherwise be an orthographic projection. Since shears are also affine transformations the entire projection remains affine and thus preserves collinearity and distance ratios.

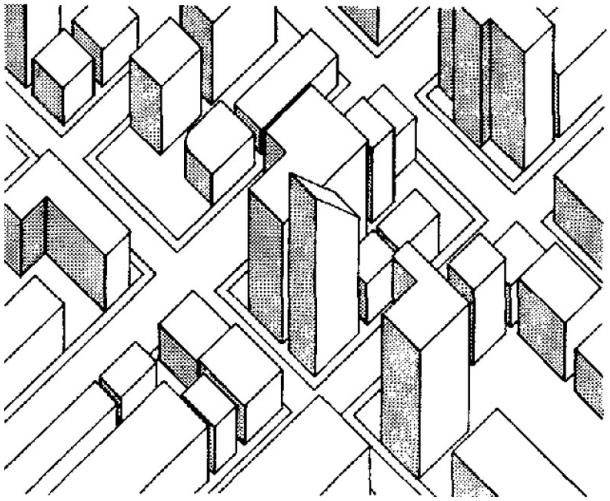


Figure 2.4: An oblique projection of a city used in a map. Images licensed for use with permission [CP78, p. 474] © ACM, Inc 1978.

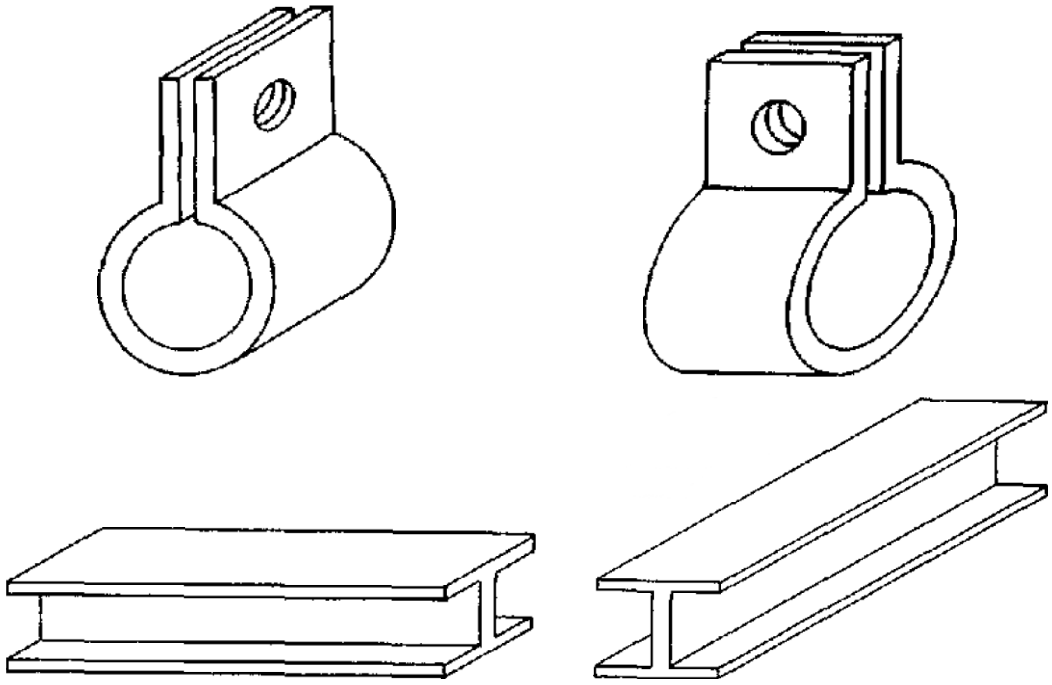


Figure 2.5: Examples of use and misuse of rules for oblique projections as described by Carlbom and Placiorek [CP78]. The images on the left show projections that follow the described rules. The top right image demonstrates the unnatural appearance produced in the scenario where the projection plane has not been placed parallel to the projection plane position; the bottom right image provides an example where the projection plane was not placed parallel to the object's principal face. Images licensed for use with permission [CP78, p. 474] © ACM, Inc 1978.

2.2.3 Perspective Projection

Perspective projection is the most commonly used projection in computer graphics. It is the same projection as is produced by a pinhole camera and approximates the projection that occurs within a human eye. A pinhole camera produces an image by allowing light to enter a dark enclosure through a small hole. Opposite the small hole is a dark wall onto which the image is projected. Figure 2.7 describes perspective projection onto the projection plane.

Unlike the parallel projections discussed previously, perspective is not suited for schematic drawings as it is difficult to determine and measure exact sizes, shapes, and angles [CP78]; what perspective does do well is present a realistic image of a scene that allows observers to visualize its three dimensional shape. Two of the key characteristics of perspective are foreshortening and the convergence of parallel lines. *Foreshortening* is the reduction in the projected size of objects that are farther from the projection plane, while the convergence of parallel lines refers to the characteristic that parallel lines that are not orthogonal to the projection plane appear to converge with one another once projected. The points at which parallel lines converge are known as vanishing points.

Two shortcomings of perspective projection are that (1) perspective portrays a monocular view [Ina91] and (2) perspective introduces angular distortion away from the center of projection (i.e., near the edges of the image) especially when dealing with wide viewing angles [ZB95]. Many nonlinear projections have been created to address these shortcomings as will be discussed in Chapter 3.

The perspective viewing volume is shown in Figure 2.1. The important attribute of the projection equation is that the x and y coordinates are inversely and uniformly scaled in proportion to depth. The equation for perspective is

$$f(x, y, z) = (\frac{x}{z}, \frac{y}{z}).$$

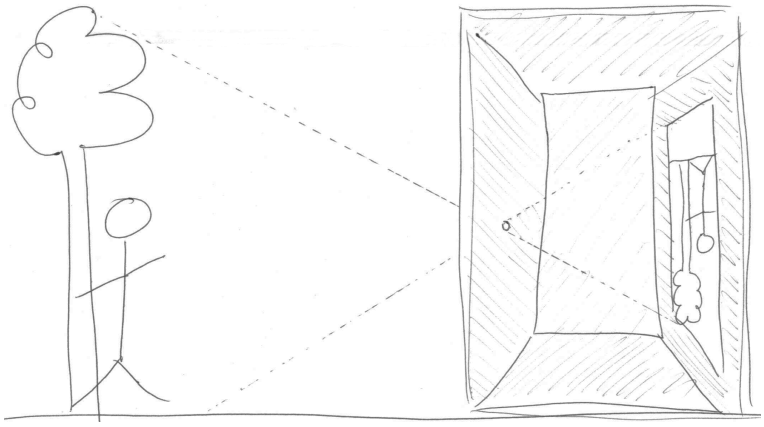


Figure 2.6: A diagram of a primitive pinhole camera. Inspired by [Hoc06, p. 245].

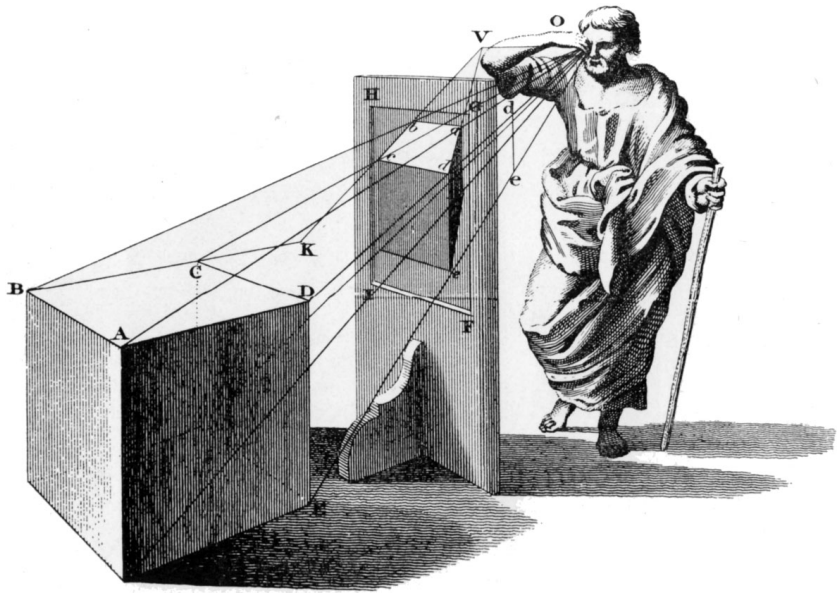


Figure 2.7: A perspective projection of a cube onto a plane. A pyramid of vision defined by the object ($ABCDE$) and the center of projection (O) is intersected by the projection plane ($FGHI$) to create an off-axis perspective projection ($abcde$). Image from [Pir70, p. 63] who acquired this figure from B. Taylor, 1811, *New Principles of Linear Perspectives.*

Using a linear formulation for depth, like those used in the parallel projection matrices leads to the matrix

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{f-n} & \frac{-n}{f-n} \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

The key difference from the previous matrices is in the last row where the 1 has been moved from the fourth column to the third column.

Within this matrix the linear mapping of depth to the range $[0, 1]$ causes problems. As illustrated in Figure 2.8, performing linear interpolation in image space often results in irregular steps across the face of the 3D triangle, this is caused by perspective not preserving distance ratios. This problem is manifested as inaccurate results in situations where scanline rendering requires interpolation of triangle attributes such as coordinates for texture mapping or lighting colours when performing Gauraud shading [Wat00]. As a consequence standard practice is to use the nonlinear depth mapping

$$z^* = \left(\frac{fn}{n-f}\right)\frac{1}{z} - \frac{f}{n-f}.$$

The derivation of this mapping is described in Appendix A and it allows correct linear interpolation between p_1 and p_2 to with $\frac{1}{p_1}$ and $\frac{1}{p_2}$.

The resulting perspective matrix is

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{-f}{n-f} & \frac{fn}{n-f} \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

With this technique to compensate for the lack of preservation of distance ratios, perspective has been integrated into the graphics pipeline. The question remains whether

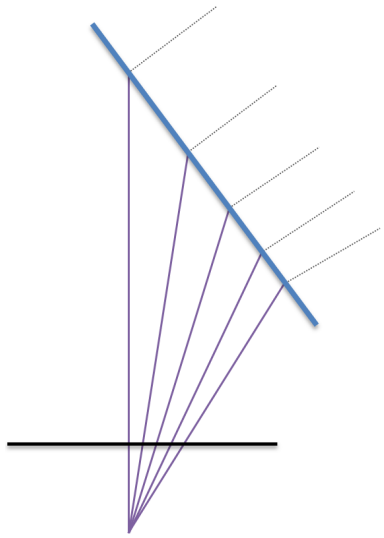


Figure 2.8: Steps taken at regular intervals on the image plane (black) create irregularly spaced steps along the face of a triangle (blue) shown in profile. Consequently linear interpolation along the image plane will yield incorrect interpolation of triangle attributes (such as texture coordinates, colours, or normals). This figure is derived from [Len04, p.117].

perspective is a linear projection. Since it is treated as such by all other literature that we are aware of [Sin02, CP78, YM04b, Wat00, Sal06], we modify our criteria of linear projections of preserving distance ratios to either preserving distant ratios or possessing an alternative depth mapping that allows for correct interpolation in image space.

The last criteria to check is see if perspective preserves collinearity. To ensure collinearity is maintained we must show that points on an arbitrary line A remain upon a line after projection. Let p_1 and p_2 be two points on A; p_1^* and p_2^* are the perspective projections of p_1 and p_2. A^* is the projection of A and we need to show that A^* is a line.

We will assume that p_1 and p_2 are within the viewing frustum since, if they are not, they will be culled from rendering processes. We can easily find points on A within the frustum by calculating the intersection between the line and the frustum clipping planes [Len04]. If no points on A are within the frustum then we do not have to render A and it can be culled.

When considering the projection of p_1 and p_2 as well as the geometry of the perspective frustum there are two scenarios: either p_1, p_2, and the eye position form a line or they do not and instead form a plane P. If they form a line, then p_1^* and p_2^* all other points along A^* will be projected to the same point and never to a curve, consequently collinearity is preserved.

If P is formed then there will be an intersection between P and the near plane in the form of another line L. Note that P and the near plane must intersect as p_1 and p_2 are within the frustum while the eye position is outside the frustum. This L naturally includes p_1^* and p_2^* as well as all other points on A^*, consequently we can be assured A^* is a line and that collinearity is maintained by perspective projection.

2.2.4 Inverse Perspective Projection

Inverse perspective is an relatively uncommon type of projection operates something like perspective but with reversed foreshortening; that is, objects become larger as they get further away.

While inverse perspective projection is rarely used, it can be found in Byzantine art (Figure 2.9) as well as in traditional Chinese landscapes. Inakage [Ina91] provides a nice scenario where inverse perspective captures an aspect of the human visual system that perspective does not. That is, when we place our eyes very near to a small box, such as a matchbook, we see the sides of the box diverge. Thus inverse perspective captures an aspect of the human visual system that is missed by perspective as well as many other types of projection.

Geometrically the viewing volume of an inverse projection is, like perspective, a frustum. However, unlike perspective, the frustum is reversed with the eye position behind the far surface as shown in Figure 2.10.

Figure 2.9: The *Old Testament Trinity Icon* by Andrei Rublev, c1410 provides an example of inverse perspective. The effects of inverse perspective are most noticeable in the shape of the chairs, the steps and the table.

The equation for inverse perspective is

$$f(x, y, z) = \left(\frac{x}{e - z}, \frac{y}{e - z}\right)$$

where e is the z component of the eye position (assumed to lie on the Z-axis). Compared to the perspective projection equation where the inverse of z is used to scale the x and y coordinates, this equation scales by $e - z$ causing essentially the same nonlinear depth mapping but oriented in the opposition direction.

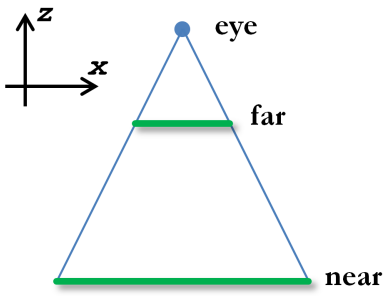


Figure 2.10: A top-down diagram of an inverse perspective projection.

This resulting matrix is

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & A & B \\ 0 & 0 & -1 & e \end{bmatrix}.$$

Inverse perspective, just like perspective, does not preserve distance ratios. Thus A and B in the matrix are the terms derived in Appendix A.1 used to create a nonlinear depth mapping that allows interpolation of triangle attributes in inverse perspective projections. The necessary depth mapping for inverse perspective to achieve this is

$$z^* = \left(\frac{e-f}{e-n}\right)\frac{1}{z} + \frac{f-e}{f-n}$$

where n and f are the perpendicular distances between the eye and the near and far planes respectively.

Next we can examine whether inverse perspective preserves collinearity. As noted, the volume of the inverse perspective is a frustum, thus the geometry based argument for collinearity in perspective projection equally holds for collinearity in inverse perspective projections.

As inverse perspective meets all three of our modified linear projection requirements we can consider inverse perspective to be a linear projection.

2.2.5 Pushbroom Projection

The projections in this and the next subsection are examples of nonlinear projection that can neither can be represented as matrices nor do they preserve collinearity. Since collinearity is not preserved we already cannot make use of linear interpolation, thus whether these projections maintain distance ratios is irrelevant.

The pushbroom projection, described by Gupta and Hartley [GH97], is created when the near surface is a horizontal line and the far surface is a rectangle with a width

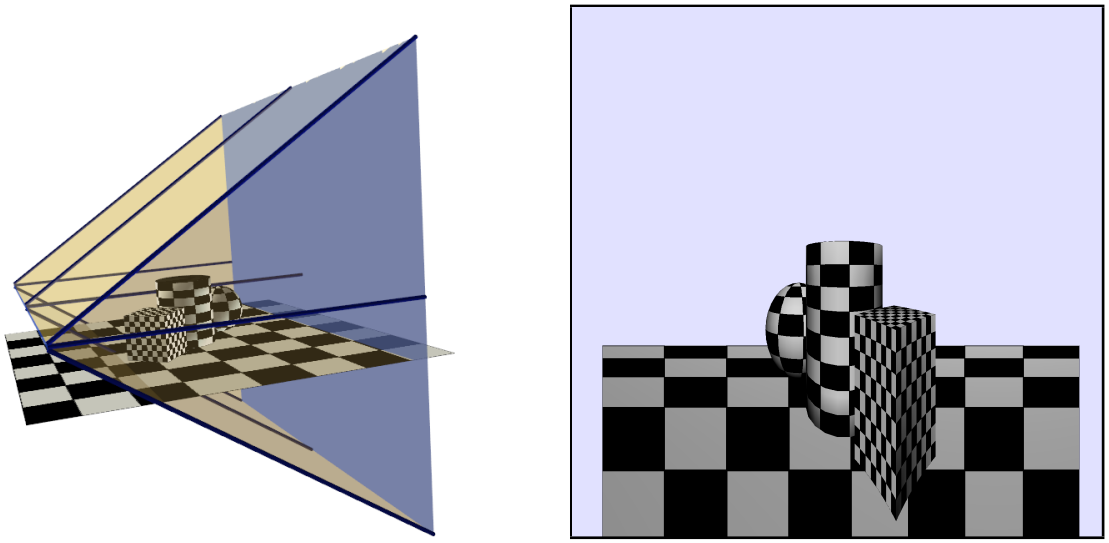


Figure 2.11: The geometry of a pushbroom projection's viewing volume (left) and the resulting projected image (right).

equal to the line as shown in Figure 2.11. This projection is used to model slit cameras (i.e., a camera that moves along a straight line). Gupta and Hartley specifically use the pushbroom projection to model the imagery captured by satellites [GH97].

A projection equation for simple pushbroom projections is

$$f(x, y, z) = (x, \frac{y}{z}).$$

Examining this equation we can see that it is some sort of mixture between a parallel projection and a perspective projection as the x component is that of an orthographic projection while the y component is that of a perspective projection. Despite these similarities it is not possible to represent this projection as a 4×4 projection matrix. In order to obtain a y coordinates with z as the denominator the resulting homogeneous coordinate should be z. Then in order for the x coordinate to not contain z the matrix must be able to produce the term xz which is not possible.

To show that collinearity is not preserved we will provide a counter-example. Consider the points $(2, 2, 1)$, $(3, 3, 2)$, and $(4, 4, 3)$ that lie upon the line containing the vector $(1, 1, 1)$ and the point $(0, 0, 1)$. The projections of the three point are $(2, \frac{2}{1})$, $(3, \frac{3}{2})$, and

$(4, \frac{4}{3})$. The differences between these points are:

$$\begin{aligned}(3, \tfrac{3}{2}) - (2, \tfrac{2}{1}) = &\quad (1, -\tfrac{1}{2}) \\ (4, \tfrac{4}{3}) - (3, \tfrac{3}{2}) = &\quad (1, -\tfrac{1}{6}).\end{aligned}$$

From these differences we can see the the projected points no longer lie along a single vector, consequently our pushbroom projection equation does not preserve collinearity. Since pushbroom projection can neither be represented as a matrix nor preserve collinearity we can safely conclude that pushbroom projection is a nonlinear projection.

2.2.6 Cylindrical Projection

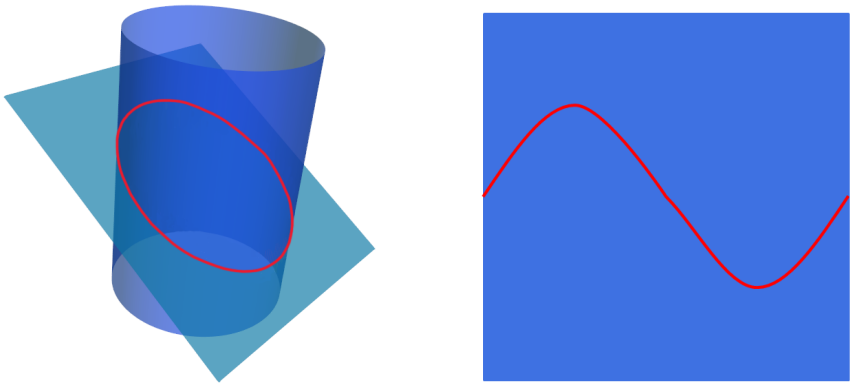
Cylindrical projections, also called cylindrical panoramas, are the result of projecting onto the sides of a cylinder with an eye position at the center of the cylinder. Cylindrical projections are used to present 360 degree views around a single axis, i.e. the type of view you would get if you stood at a fixed position and turned in place to see your entire surroundings.

The simplest projection equation for a cylindrical panorama where the axis of the cylinder corresponds to the y-axis is

$$f(x, y, z) = (\arctan \frac{x}{z}, \frac{y}{z}).$$

It is clear that due to the presence of the trigonometric function tan that it is not possible represent this transformation as a matrix.

In order to determine whether cylindrical projections preserve collinearity it is easiest to examine this projection geometrically. Given an arbitrary line we can construct a plane that contains both the line and the eye position inside the cylinder. The elliptical intersection between the plane and the cylinder identifies where the line will be projected as well as positions on a cylinder where other lines contained within the plane would be projected. When the cylinder is unwrapped to create a rectangular image the ellipse



(a) (b)

Figure 2.12: Intersection of a plane and a cylinder. In (a) the ellipse formed by the intersection has been coloured red. (b) Unrolling the cylinder with the marked ellipse to produce a rectangular image shows that the ellipse forms a curve. Consequently, any line that originally lay upon the plane is projected to a curve by a cylindrical projection.

becomes a curve illustrating that lines become projected to segments of this curve and consequently that cylindrical projections do not preserve collinearity. There are two cases where collinearity is preserved, the first is when the aforementioned plane is perpendicular to the cylinder's axis, the intersection becomes a circle that is unwrapped to create a horizontal line. The other is when the plane is parallel to the cylinder's axis, creating a vertical line when the cylinder is unrolled.

2.3 Summary

In this chapter the differences between linear and nonlinear projections have been outlined. Linear projections can be represented by matrices in homogeneous coordinates, preserve collinearity and either preserve distance ratios after projection or the changes in the distance ratio can be compensated for (as with perspective). Any projection that fails any of these requirements is considered nonlinear. This distinction is necessary to determine which projections are linear and consequently can be easily integrated into the standard graphics scanline renderings routines.

Chapter 3

Existing Nonlinear Projections and Related Work

There exists a wide variety of literature describing, analyzing, and making use of nonlinear projections dating back to the Renaissance and earlier. The goal of this chapter is to provide an overview of this work so that it may be compared to Flexible Projection. To limit the scope of this thesis, this chapter concentrates on works in the field of computer graphics that describe the creation of different types of nonlinear projections. We have separated this literature into the following categories: nonlinear projections created by *projection onto non-planar surfaces* (Section 3.1); *camera models* that feature nonlinear projection (Section 3.2); *lenses* describe projection techniques that introduce local areas of alteration within perspective projections (Section 3.3); *cartographic projection* (Section 3.4); *implicitly defined projections* are those that, instead of being defined explicitly, are defined by an underlying dynamical system (Section 3.5); *multi-camera projections* are created by combining or otherwise blending images from several linear or nonlinear projections (Section 3.6); and the last category is *camera-based deformations* that blend camera oriented control with the modeling process of deformation (Section 3.7). Later, in Chapter 6 we will explicitly describe how these nonlinear projections can be reproduced with Flexible Projection.

3.1 Projection onto Non-Planar Surfaces

A wide variety of nonlinear projections are made possible through the use of curved projection surfaces.

Inakage [Ina91] introduces a type of curvilinear perspective that is essentially a projec-

tion onto a segment of a sphere. The described projection equation includes a parameter controlling the offset between the eye position and the projection surface that allow the viewing angle to be changed. Perspective and inverse perspective are also discussed. Additionally 3D warping is used to manipulate the distance parameter that controls foreshortening (and inverse foreshortening) as well as to distort the entire scene before projection.

Levene [Lev98] incorporates the projections used by Inakage's into a single interactive system, developing a projection equation that transforms a 3D point P into a 2D point P^*:

$$P^* = \begin{pmatrix} C_x + [f(P) \cdot V_x] \\ C_y + [f(P) \cdot V_y] \\ P_z \end{pmatrix}$$

where the center of scaling C is an image space point, V is the vector $(P_x - C_x, P_y - C_y)$, and f is a function that determines the scaling factor for the components of P relative to C. In practice C is used to control the location of vanishing points in the image and can be varied with depth to force parallel receding lines to follow curved paths to vanishing points. The function f, determined by a projection surface, varies over the image and controls the amount of convergence/divergence between orthogonal lines. The projection surface can be manipulated into concave and convex shapes, allowing various curvilinear projections to be created. Figure 3.1 shows two images projected by Levene's system.

In standard ray tracers the pixels are traced by placing the ray origins on the image plane and then specifying ray direction based on perspective or parallel projection. There are a variety of techniques that describe different approaches to specifying ray origin and direction. In Optical Models [WM90] projections are described by defining the near projection surface as well as direction for the projectors. Mathematically this is a mapping from two dimensions to five (e.g., the two dimensions of the image to the three

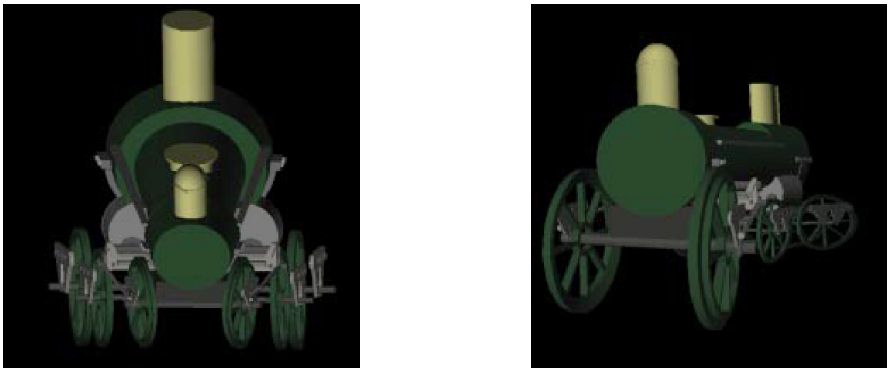


Figure 3.1: Two projected images created with Levene's nonlinear projections. The left image is an example of inverse perspective; the right image shows a projection that causes parallel receding lines to follow a curved path. Images used with permission [Lev98, p. 39] © Levene, 1998.

dimensions of the near projection surface and the two angles required to describe the direction of the projectors). Wyvill and McNaughton suggest that this provides a useful tool in terms of flexibility due to Optical Model's description by a mathematical function, and in terms of ease of implementation as these projections can be easily integrated into ray-tracing systems. Distortion Cameras [AG95] suggest the use of surfaces such as cylinders, spheres and free-form surfaces to provide ray origin. Change in ray direction over the image plane is specified with nonlinear control functions. An example provided is the specification of centers-of-interest in the scene that change ray direction or image surface to increase the presence of these centers of interest in the produced image.

Digital Cubism [Gla00, Gla04a] creates projections by defining two non-uniform rational B-spline (NURBS) surfaces. The first surface is the near projection surface, the second is used to provide direction to the projectors, thus each projector is defined by:

$$p_{u,v}(t) = S_1(u,v) + t(S_2(u,v) - S_1(u,v)) \quad t > 0 \tag{3.1}$$

where S_1 and S_2 are the NURBS surfaces and u, v denote parameters within the parametric range of the surfaces that can be mapped to image coordinates. In a more advanced interface [Gla04b] the image creator provides pieces of image from existing projections

Figure 3.2: Left: creation of a digital cubist image by selecting three image segments. NURBS surfaces are created that capture these segments and interpolate between them. Center: the interpolated areas. Right: the resulting image. Images used with permission [Gla11] © Glassner, 2011.

places them where they are desired in the resulting image and then ray origins and directions are algorithmically interpolated to fill in the blanks. Glassner makes use of these projections to explore possibilities in image creation and animation that go beyond traditional perspective projection. For instance the projection shown in Figure 3.2 is able to capture the facial expressions of two people in a face-to-face conversation within a single image.

Trapp and Döllner's Single-Center Projections [TD08] all feature a single point that acts as the near projection surface (e.g., a position where all projectors originate). To provide direction to the projectors, the image creator must provide a normal map. This normal map is essentially a vector field in image form. The authors suggest the use of a set of normal map tiles that describe commonly used projections (e.g., perspective, cylindrical panorama, etc). These tiles can then be combined in various configurations to easily create new projections. Trapp and Döllner also describe use of a discontinuous normal map for the creation of multi-camera type projections (see Figure 3.3 for an example).

To summarize, all of these projections onto non-planar surfaces examine differing approaches to specifying projection surfaces and projectors, providing a very similar approach to the one we have taken with Flexible Projection. The main difference between

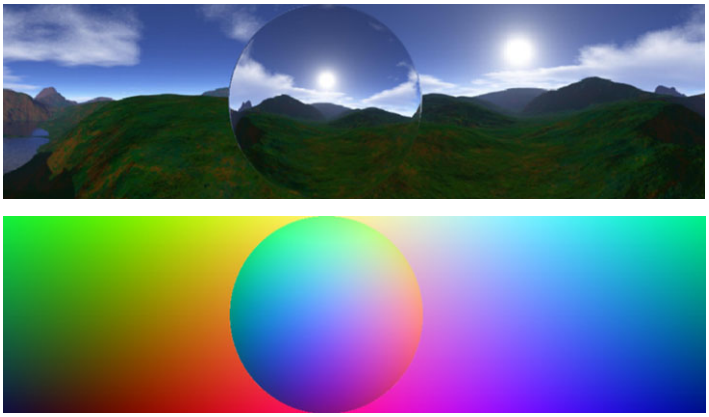


Figure 3.3: Creation of a single-center panorama with a discontinuous normal map. Top image is the created projection, bottom image is the normal map used to create the projection. Image licensed for use with permission [TD09, p. 60] © Springer, 2009.

them is that Flexible Projection accommodates a wider variety of possible projections, particularly through allowing projectors to follow curved paths.

3.2 Camera Models

This section classifies a variety of nonlinear projections as camera projections. Certain projections are inextricably linked to different types of cameras. Computer vision researchers have found it useful to model many varieties of camera which in turn has lead to the description of a wide variety of nonlinear projections. Aside from determining formulations for different projections, these works often emphasize computer vision applications where the determination of camera parameters from an existing image is a primary concern. Examples of applications that have made use of projections in this subsection include:

- modeling and correcting the distortion prevalent in specific types of photographs [GH97, ZFPW03, AL07, CF05, ZB95];
- 3D reconstruction [GH97];
- embedding extra information for image based rendering [PA06, MPS05, RB98];
- for artistic effect in produced images [HCS+07];

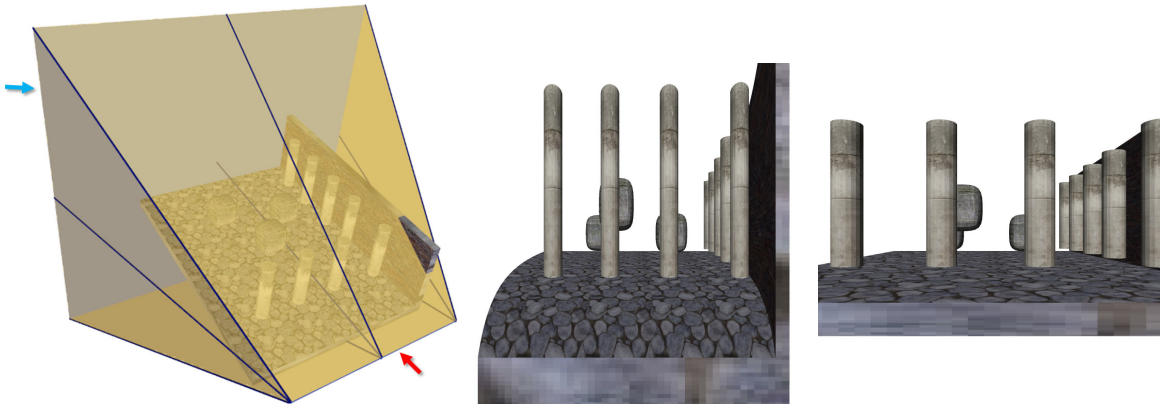


Figure 3.4: Left: viewing volume of a pushbroom projection with embedded scene. The red and blue arrows respectively identify the near and far surfaces. Center: resulting image. Right: result of perspective projection of same scene from same direction.

- determining inverse equation(s) that allow determination of camera parameters from a projected image [CF05]; and

- encapsulating other camera models within a single framework [YM04b, HCS+07].

The remainder of this section describes these techniques individually.

Pushbroom camera projections, previously mentioned in Section 2.2.5 describe the situation where a camera capturing a perspective column of pixels is moved along a straight line orthogonal to the view direction (i.e., sideways). As the camera moves to the side, another column of pixels is captured and appended to the previous column. This configuration results in the projection not having an eye position, rather the projectors converge to a line, the line along which the camera was moved. An example of a pushbroom projection image is shown in Figure 3.4. Gupta and Hartley [GH97] describe the use and accuracy of Pushbroom projections for analyzing satellite imagery.

The Crossed-Slits projection described by Zomet et al. [ZFPW03] provides another model for slit cameras where all projectors intersect two lines (rather than the one line used in pushbroom cameras). The authors argue that the images produced by crossed-slits are perceptually closer to perspective images. This technique can be used for image

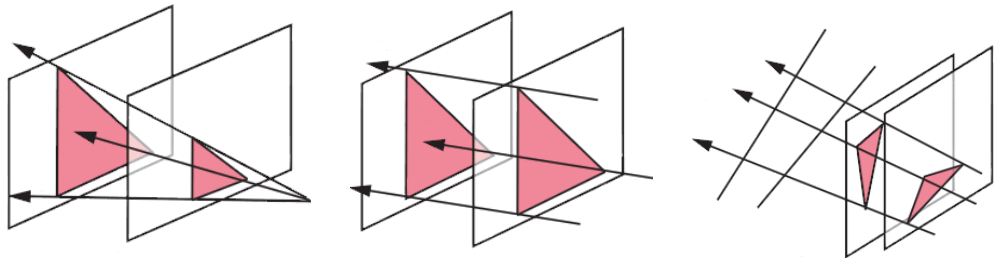


Figure 3.5: Three GLCs defined by the relative positions of two triangles. Left: recreation of perspective projection. Middle: recreation of orthographic projection. Right: recreation of a crossed slits projection. Images licensed for use with permission [YM04b, p. 19] © Springer 2004.

based rendering through resampling a crossed-slit image to create a perspective image from a variety of viewpoints, allowing for virtual camera movements.

The Rational Function Lens Distortion Model [CF05] is able to estimate and correct the radial lens distortions created by wide-angle lenses. This model determines a particular physical lens' distortion parameters given a single calibrated image or two uncalibrated images of the same scene. This technique operates by defining a mapping between image coordinates and projector rays as a combination of nonlinear functions of images coordinates, allowing their developed linear algorithm to estimate the nonlinear projection. In this fashion this technique can describe projections created by physical camera lenses that differ from true perspective images that are often the photographic ideal. Common lens projections that can be handled include fisheye, pin-cushion, and barrel projections.

General Linear Cameras (GLCs) [YM04b] provide a common framework for several different cameras, including pushbroom [GH97], cross-slit [ZFPW03], perspective, and parallel projections. Individual GLCs are specified by defining the corners of two triangles that have the same normal and are not coplanar (as shown in Figure 3.5). The development of GLCs provided insight that led to the discovery of three new camera types: twisted orthographic, pencil, bilinear cameras. Hou et al. [HWSG06] describe a

technique whereby GLCs can be rendered in realtime with graphics shaders on the GPU. Note that although GLCs can be used to produce nonlinear projections (such as pushbroom [GH97] cameras), the authors term this model "linear" because GLCs are defined through an affine combinations of lines. Adams and Levoy [AL07] further generalize GLCs to incorporate focus and depth of field.

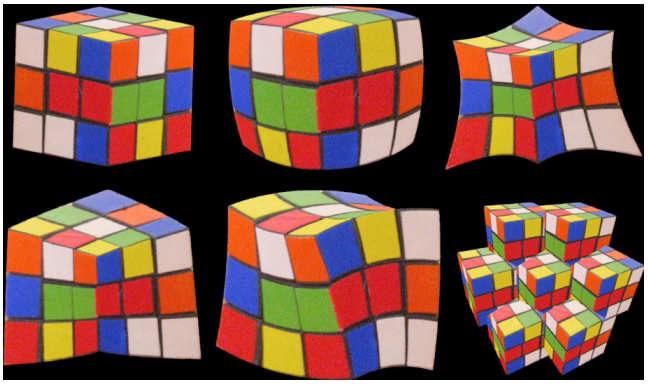


Figure 3.6: RTcam projections created from a picture of a Rubik's cube. From left to right the top row of images demonstrate perspective, barrel and pin-cushion projections; the bottom row provides examples of inverse perspective, a depth-dependant twist, and a compound lens effect (i.e., a fly's eye-like projection). Image used with permission [HCS+07, p. 967] © 2007 IEEE.

RTCams [HCS+07] use tensors to perform projections, producing an effect that is equivalent to a geometric mapping followed by a perspective projection. Much like the way matrices map lines to lines, tensors can map lines to quadratic curves. In their system the authors suggest decomposing these tensors into three parts: scalar, linear, and quadratic. The linear and scalar parts can be used alone to recreate any linear projection. The quadratic component of the tensor is manipulated to produce nonlinear projections. This is one of the few techniques aside from Flexible Projection where the projection can be organized to change through the depth of the projection. An example would be a projection that is orthographic at near surface and then transitions to perspective near the far surface. The authors show that RT cameras can reproduce

many other cameras including GLCs, pushbroom, cross slits, rational function cameras, and strip cameras [HCS+07]. Examples are shown in Figure 3.6.

Multiple-Center-of-Projection (MCOP) Images [RB98] is a technique for creating a single image that is acquired from many points of view. The underlying idea of the projection is to imagine a camera being moved along a path and at each point on the path the camera contributes a column of pixels to the final image; essentially similar to a pushbroom, but without a fixed viewing direction and following a curved path. These images are able to portray elements of the scene in more than one position in the final image.Rademacher and Bishop also demonstrate how these images can be used for image based rendering. A technique that Rademacher and Bishop claim to be a subset of MCOPs that also captures a single image from a camera path is that of Multiperspective Panoramas [WFH+97]. Wood et al. [WFH+97] develop panoramas for cel animations that incorporate both continuous changes in camera position specifically including panning, tilting, trucking, and zooming.

The Occlusion Cameras [MPS05] is an example of a nonlinear projection where the behaviour of the projection changes through the depth of the projection volume. This projection was developed for image-based rendering to allow camera movements to nearby viewpoints, providing means to encode and capture nearly visible areas of an object that would be occluded in a perspective projection. Perspective and occlusion camera images act as pairs that can be used to synthesize perspective images from nearby viewpoints.The occlusion camera projection volume is defined by a pole, two planes, and a magnitude of distortion. The pole is a vector from the eye position through the centroid of the occluder.The two planes mark where the volume differs from a perspective projection volume. The first plane, z_1, marks where the distortion begins (slightly in front of the occluding object) and the second is usually coincident with the far projection plane. The volume of the occlusion camera projection from the viewing position to z_1 is that of a

perspective projection. The remainder of the volume is contracted causing projectors to bend inward proportional to depth so as to capture the nearly visible areas on the model. This bending inward is accomplished with a 3D radial distortion function.

The Depth Discontinuity Occlusion Camera [PA06] has the same purpose in assisting image based rendering but is aimed at capturing occluded parts of relatively complex scenes (rather than single objects). These projections are achieved with a piecewise bend of the projectors that pass near occluding objects in order to sample regions of the scene that have been occluded. The alteration of projectors from lines to polylines is controlled by a distortion map. This distortion maps encodes the direction and depth of the distortion and is calculated from the depth buffer information in a reference perspective image.

An interesting point is that neither occlusion nor depth discontinuity cameras can be reproduced by RTCams [HCS+07] or other camera-based frameworks. This highlights the inabilities of cameras based frameworks to include projections that make use of piecewise or curved projectors.

Projections from the Camera Model category of nonlinear projection all demonstrate strong ties to treatment of projections created by physical cameras and lenses and tend to be strongly embedded in mathematical analysis of the created imagery. In comparison to Flexible Projection these works seldom discuss rendering 3D virtual scenes (i.e., capturing images), instead most of these techniques are designed for image-based rendering where images are converted from one representation to another (e.g., from pushbroom to perspective).

3.3 Lenses

In this group of work, images similar to those created by nonlinear projections are specified with localized distortions. Several of these techniques are performed in image space, making it unclear whether or not they should be truly considered to be projections; however since these techniques can be used to reproduce specific projections (such as fisheye projections) it seems relevant to include these works.

In Magic Lenses [YCB05] nonlinear projections are implemented by performing a 2D mapping on a 2D image that corresponds to the situation where a lens (a curved, transparent surface) is placed in front of the projection plane.The authors argue that these deformations incorporate depth, view angle, and camera position, in a way that straightforward 2D image deformation does not. Several spherical and cylindrical lenses are demonstrated.

The Elastic Framework [CM01] envelopes a large number of detail-in-context lenses that provide distortion, magnification, and detail-in-context presentations. Explicit control is provided over magnification, position, focal shapes, folded status (i.e., whether the lens is viewer aligned or not), drop-off functions, and distance metrics. Example lenses are shown in Figure 3.7. The lens effects are implemented by 3D deformation of the 2D data. The resulting 3D points are then projected back to the image plane with perspective projection.

Carpendale et al. [CCF97] and Cowperthwaite's thesis [Cow00] provide a kind of lens that maintains non-occluded views of one or more specific objects (or regions) within crowded 3D environments. The non-occluded views are created through displacement based on a variety of distance metrics or displacement functions that deform the scene in a viewer aligned fashion. An example is shown in Figure 3.8. This work can also be considered a camera-based deformation (Section 3.7).

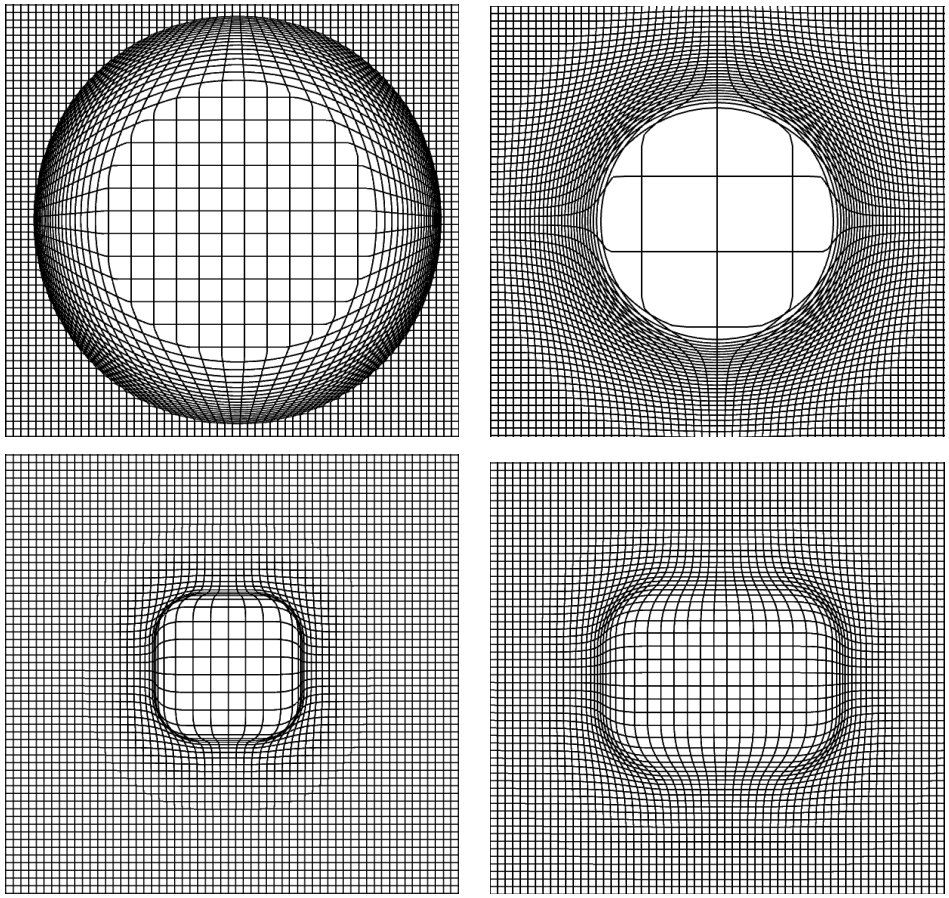


Figure 3.7: Several lens types from the Elastic Presentation Framework [CM01] applied to a regular grid of lines.

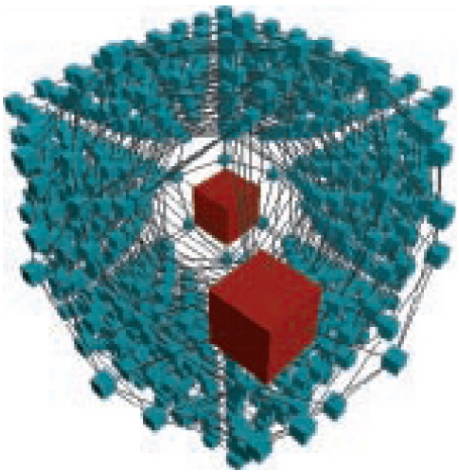


Figure 3.8: Distortion viewing of a cube filled with equally sized cubes. In this image two areas of focus (red) have been specified. Viewpoint based displacement is used to ensure visibility and magnification provide emphasis of focus area detail. Image used with permission [CCF97, p. 48] © 2007 IEEE.

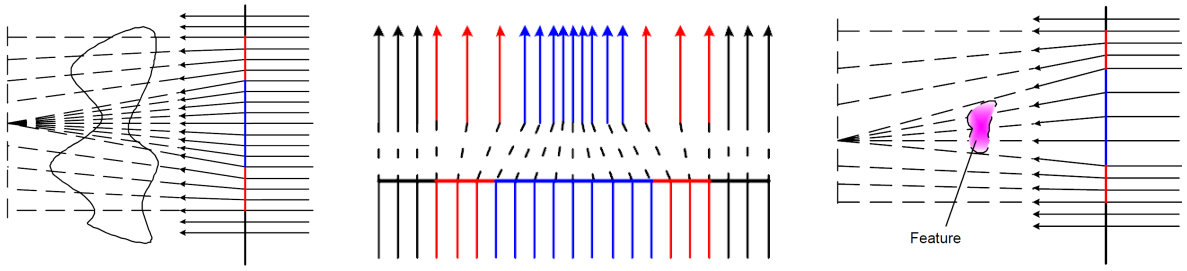


Figure 3.9: Schematics for the three Magic Volume Lenses [WZMK05]. Left: the magnifier lens bends rays to point toward a specific point. Center: the sample rate lens alters the distribution rays, compressing them in the area of interest. Right: the feature based lens operates much like the magnifier lens but, instead, bends all the rays necessary to cover the specified object/feature. Images used with permission [WZMK05, pp. 48, 50, 51] © 2005 IEEE.

Magic volume lenses [WZMK05] operate within orthogonally projected, ray-casting, volume rendering environments. Three lenses are discussed that provide local magnification through different specifications. The first magnifies by moving the direction of affected rays inwards, another concentrates the rays by altering their distribution. The last lens type causes magnification of a specific feature or object and only bends the ray necessary to magnify the object. This is accomplished using two passes of rendering where the first pass establishes which rays should be altered. A schematic comparison of the three lens types are shown in Figure 3.9.

Overall, in comparison to Flexible Projection these lens-based nonlinear projections concentrate on 2D or localized changes to imagery. Flexible Projection can support these kinds of local changes within the viewing volume.

3.4 Cartographic Projections

Perhaps the most commonly known nonlinear projections are those created by cartographers to portray the Earth as a flat surface with a minimum of distortion. While these projections were not developed exclusively for computer graphics, computers are commonly used to accurately and uniformly implement these nonlinear projections.

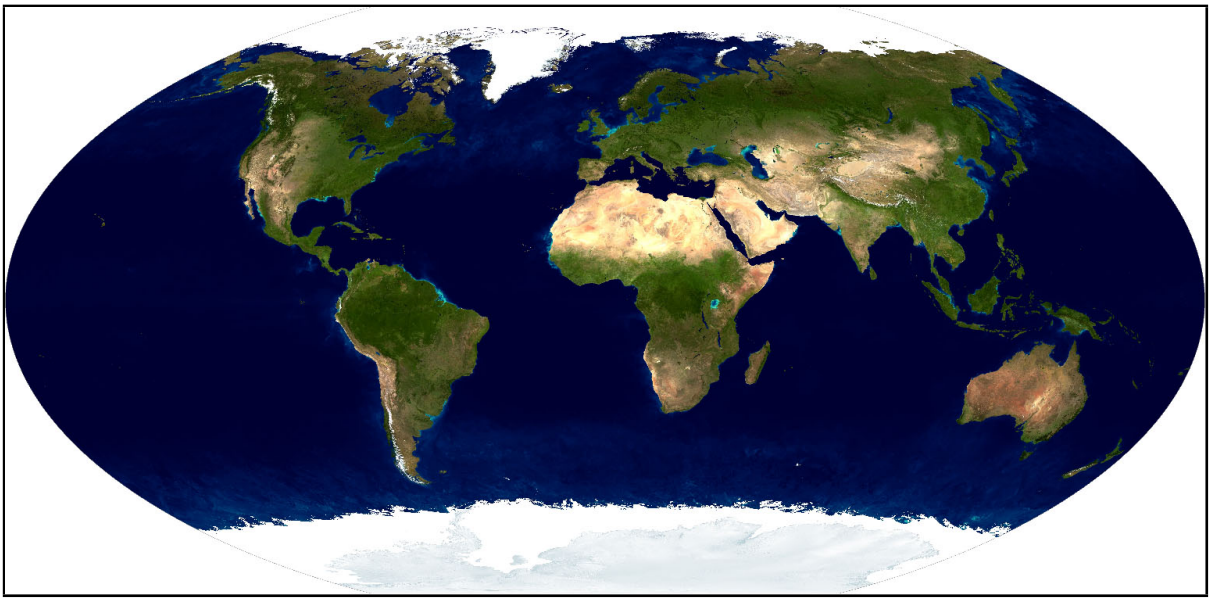


Figure 3.10: A Winkel Tripel projection of the earth. Satellite imagery from [NAS02].

As noted by Salomon, "Any attempt to open, unfold, or unroll a sphere to lie flat results in stretching and deforming it in some way." [Sal06, p. 199]. In general cartographic deformations affect measurement of some or all of: angle, area, scale, and shape. It is often possible to preserve one or two of these properties at the expense of others. For instance azimuthal projections preserve shape, angle, and distance relative to a central point on the map. The Mercator Projection, developed by Gerhardus Mercator in 1569 [Sal06], preserves angles and shape while introducing deformations to area and scale. The Winkel Tripel projection (Figure 3.10) while not exactly preserving area or angles was designed to minimize the distortion of area, angle, and distance [Fur97].

There is a vast array of literature on this subject, however Snyder [Sny93] and Furuti [Fur97] provide current, useful, and extensive overviews. While cartographic projection are nonlinear transformations they are essentially 2D to 2D transformations (e.g., the surface of a spheroid mapped to a planar surface) rather than the 3D to 2D projections that are discussed in terms of Flexible Projection.

3.5 Implicitly Defined Projections

Another grouping of nonlinear projections are those defined implicitly by underlying dynamical systems. Such projections are not explicitly defined ahead of time; rather they are defined by a dynamical system and chosen starting points. The projection definition is derived by integration of a vector field. This integration results in projectors that follow curved paths, consequently these works are described as nonlinear ray tracers.

Gröller's work [Gr5] concentrates on nonlinear curves that result implicitly from an underlying dynamical system. The curves become solutions to first order differential equations with given initial values. The curved rays are traced by iterative local approximations with line segments. The approximations are made using Euler or Runge-Kutta methods. Gröller also mentions possible efficiencies in tracing explicitly defined parametric curves. The described technique uses a hierarchical structure of axis-aligned bounding boxes that subdivide the curve until curve segments can be accurately approximated by line segments [Gr5]. Interactive versions of these systems allow placement of gravity attractors (with local influence) that can be used to provide indirect control over the projection.

Similar to Gröller, Weiskopf uses Runge-Kutta integration to ray-march four dimensional rays, providing educational visualization of gravitational physics [Wei00, WSE04]. An example is shown in Figure 3.11. Stam and Languenou produce linear approximations of rays that are deformed by non-constant media such as air [SL96].

While these implicitly defined projections and Flexible Projection are both capable of creating projections with nonlinear projectors, the approach taken in Flexible Projection is to define these curved projectors explicitly rather than relying on vector fields to create the volume. This difference in approach provides Flexible Projection with precise definitions for projectors (rather than concern over curve accuracy when integrating through

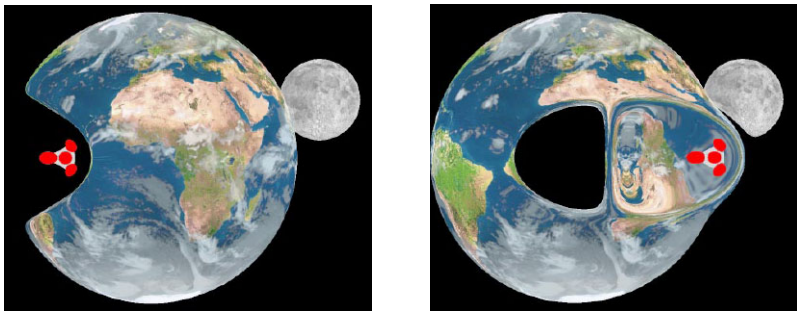


Figure 3.11: Visualization of a warp bubble, a physically-plausible, theoretical method of traveling faster than light produced by Weiskopf's nonlinear ray tracing system. Image used with permission [Wei00, p. 449] © 2000 IEEE.

a vector field) and allows use of foreknowledge of the curve's path to speed ray casting intersection tests.

3.6 Multi-Camera Projections

Multi-camera projections are projections that combine several individual projections into a single image. Usually the combined projections are linear projections but this does not always have to be the case. Often multi-camera projections are used to discretely approximate other nonlinear projections. A common example is that of cylindrical panoramic projections created by stitching together several photographs taken from different orientations of the camera at the same position. This scenario is one of many that creates a nonlinear projection through a multi-camera technique. Other multi-camera projections are only suited to projection that feature discontinuous mixing of different projections. For instance, a collage image where the projection of an object is placed on top of an image created from a different projection.

Collomosse and Hall [CH03] collage together pieces of several images to create cubist style renderings. An example is shown in Figure 3.12. Automating Joiners [ZMP07] provides another collage-like technique where pictures taken from different positions or directions are stitched together.

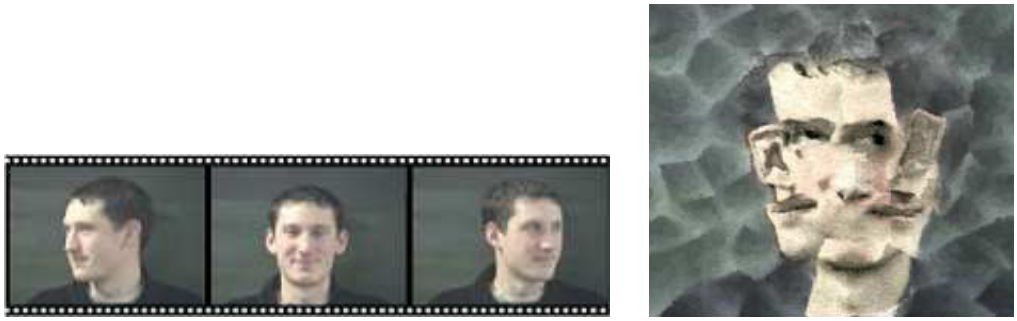


Figure 3.12: Input images (left) used to create a cubist style image (right) with Collomosse and Hall's cubist style rendering algorithm [CH03]. Image used with permission [CH03, p. 451] © 2003 IEEE.

Agarwala et al. [AAC+06] use photographs taken over long scenes (e.g., streets, grocery aisles, river banks, etc.) to create multi-viewpoint panoramas. The photographs are gathered using a hand held camera that takes pictures at relatively even intervals along the scene. Optimization is used to select elements from overlapping images to ensure all parts of the scene are shown straight on while minimizing seams. This ensures that the pixel color matches that of the overall average representation thus compensating for transitory effects such as specular highlights.

Levene [Lev98] discusses the combination of separate projections into a single image. The projections' images are merged using 2D inertial fitting where one projection is arbitrarily selected as the default projection and all other projections are merged into the image. The merging is accomplished by fitting: position through aligning the images' centroids; orientation through aligning the image's axes of inertia; and lastly size through scaling to fit a 2D bounding circle. This technique suffers from problems in determining correct visibility and as well as merging orientation when there is a lack of unique axes of inertia.

Agrawala et al. [AZM00] introduce a multi-camera system where 3D objects can be individually linearly projected and then composited into a single image. Images are created by combining the images in 2.5D normalized device coordinates. Depth

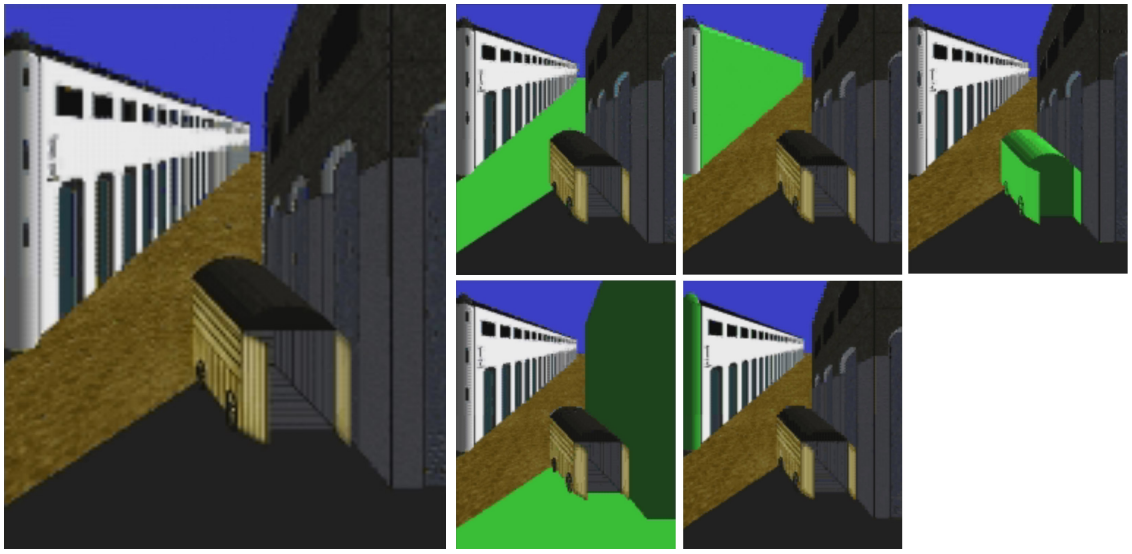


Figure 3.13: Left: a recreation of Giorgio de Chirico's *Mystery and Melancholy of a Street* with Aristic Multiprojection [AZM00]. Right: the individual camera views that were combined to create the multiprojected image as well as a top-down schematic of the scene's components. Images used with permission [AZM00, p. 134] © Springer-Verlag 2000.

ordering and visibility are dictated by depth measurements from a user-chosen master camera. This work also presents a collection of constraints upon perspective projections to maintain an object's size, a fixed view direction, a fixed position, or a straight on view orientation. Examples are provided demonstrating that this technique can be used to imitate a variety of art works, including paintings by Cezanne, Paolo Uccelo, Raphael, and Giorgio de Chirico. An example recreation of an artwork with this system is shown in Figure 3.13.

Singh [Sin02] introduces a multi-camera system where the camera settings can be linearly blended across the scene with a system of weights. Singh suggests weighting the influence of each camera based on 3D position relative to a camera's center of interest or a directional weighting scheme that bases weights upon each camera's viewing direction. After the weights have been calculated they are normalized and used to interpolate camera settings to individually project each vertex in the scene.

Coleman and Singh's RYAN system [CS04] extends Singh's previous work [Sin02] by making use of boss and lackey cameras. The boss camera provides general spatial position for all the elements of the projected scene. The lackey cameras provide distortions of areas of the scene such that, when viewed from the boss camera, the objects appear as if projected by the lackey camera but translated in image space as if projected by the boss camera. Weighting schemes for lackey cameras can be formulated as in Singh's previous work [Sin02]. The authors introduce constraints that assist in preserving the relative positions and sizes of objects in complex scenes. The described system was used by animators to create the animated documentary *Ryan* [Lan04]. This work has been further expanded: to automatically generate projections based on datamining a user's exploration of the scene [SB04]; to develop a tool-kit of widgets that automatically place the necessary cameras for often-used effects (e.g., unwrap, clipping, fisheye, and panorama) [SGS08].

An additional interesting point in the literature that Coleman and Singh [CS04] address is the issue of how to perform lighting in a nonlinear projection. That is, should lighting and shadow calculation occur in 3D before projection, or afterwards based on the geometry and normals in 2.5D after projection? The authors point out that since they are not actually distorting the geometry that shadows and lighting should be entirely based on this underlying geometry and not the deformed geometry seen in the projected image. An additional issue is that diffuse and specular lighting calculations require a unique viewing direction, problematic with multi-camera (or nonlinear projectors). The authors suggest two alternatives: using the viewpoint of the interpolated camera; or averaging illumination based on the weight contribution of each camera. The authors note that animators using their system preferred the second approach as it provides better predictability. An example is shown in Figure 3.14.

RTCams [HCS+07], mentioned in Section 3.2, can also be combined serially or in

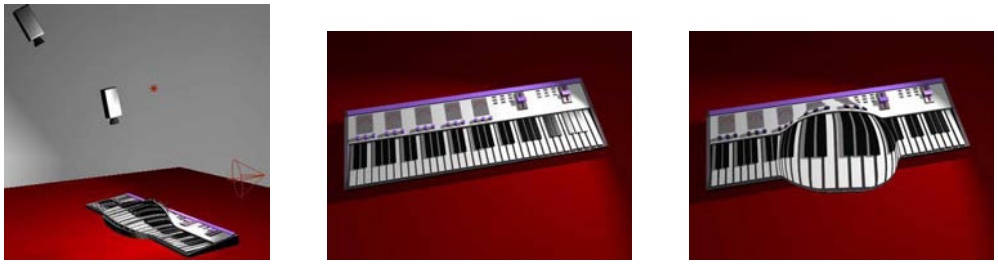


Figure 3.14: Creating a multi-camera projection with Coleman and Singh's RYAN system [CS04]. Left: camera positions as well as distorted scene geometry. Center: the original image as seen from the master/boss camera that is used to provide lighting and position. Right: the resulting multi-camera projection. Images licensed for use with permission [CS04, p. 134] © ACM, Inc 2004.

parallel. Combination in parallel is achieved similarly to Coleman and Singh [CS04] where a hierarchy of weighted RTCams is created. As RTCam tensors are difficult to work with directly, the authors suggest building a library of useful, parameterized tensors designed to produce various component projections (e.g., spherical, perspective, orthogonal, etc). This library of tensors can then be used to mix and match, applying different projections to different parts of the scene through user selection.

Yu and McMillan [YM04a] have created a multi-camera system that blends together general linear cameras (GLC) [YM04b]. Essentially this system creates a large viewing volume by placing viewing volumes of different GLC projections next to one another, touching but not overlapping. Continuity in the image is maintained by placing the individual volumes side-by-side. The paper outlines the GLCs that may be placed neighbouring one another in order to achieve this continuity. An example is shown in Figure 3.15.

Popescu et al.'s Graph Camera [PRAV09] is designed to circumvent occluders within a scene, allowing many regions of the scene to be viewable, without redundancy, within a single image while maintaining C^0 continuity of the scene in the image. The projections are created by bending, merging, and splitting the viewing volumes of individually placed perspective cameras to create a single projection volume. Feed-forward rendering is

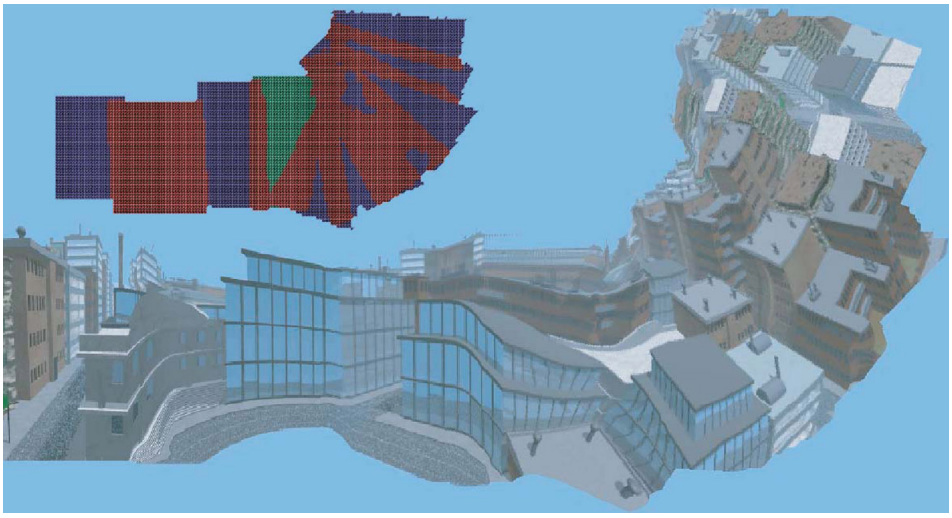


Figure 3.15: A multiperspective image created from a mixture of GLC projections. The map in the upper left provides colouring to indicate the projection used to create different parts of the image. Green indicates pushbroom projection, blue for perspective, and red for crossed-slit. Image from [YM04a, p. 68] © Eurographics Associates 2004. Reproduced by the kind permission of the Eurographics Association.

discussed as well as the use of video, allowing objects in the scene to be monitored continuously in a single image rather than jumping between multiple video feeds.

Multi-camera projections are the category of nonlinear projection most divergent from those created by Flexible Projection. The approach of multi-camera projections is to blend or mix and match several target images/projections into a single image/projection. This is opposed to Flexible Projections general approach of modeling the viewing volume as a whole, more of a top-down approach. An in-depth constrast and comparison of multi-camera projections to Flexible Projection is provided in Section 6.2.

3.7 Camera-Based Deformation

While there are many similarities between projection and deformation, the key difference is that projection allows manipulation of the camera and the perception of the scene and, consequently, is created to assist in this specific task. Deformation is a modeling

technique used to create objects of a specific shape. Another difference is that projection does not ordinarily affect lighting effects such as specular highlights and shadows. An unexplored exception to this rule exists where projections simulate the bending of light (such as in Section 3.5). A last difference is that, being based on the camera, projections affect the scene as a whole usually making it easier to achieve coherence in animated environments.

Due to these differences between projection and deformation we limit this discussion to works that specifically consider deformations based on a camera's position or orientation. Inakage [Ina91] provides an early reference to the use of deformation, discussing the use of 3D transformations before projection to 2D.

Both Martin et al. [MGT00] and Rademacher [Rad99] have described nonphotorealistic rendering techniques that deform objects based on a function of their distance from the camera. Rademacher's view dependent geometry technique [Rad99] uses a set of key deformations and corresponding viewpoints and then, given a particular viewpoint, uses interpolation to deform the model to match the viewpoint. The suggested application for this work is use in cel animation where objects are shown with specific deformations to adjust for specific viewpoints. For example, a rabbit's ears are deformed so that the viewer can always clearly see both ears but, despite rotation, never see the ears in profile. Similarly, Martin et al.'s observer dependent deformations are performed with a 1D, 2D, or 3D control function that can make use of camera position, orientation, or time (allowing for animated effects) as function parameters that control deformation.

Lorenz et al. [LTJD08] use a view-dependent deformation to recreate interesting visualizations of maps that include 3D building models. Two visualizations are demonstrated. The first provides a bird's eye view where buildings are shown from nearly straight above near the viewer at the bottom of the image and shifts to a perspective view in the distance. This is accomplished by deforming the map geometry as if it was placed on a

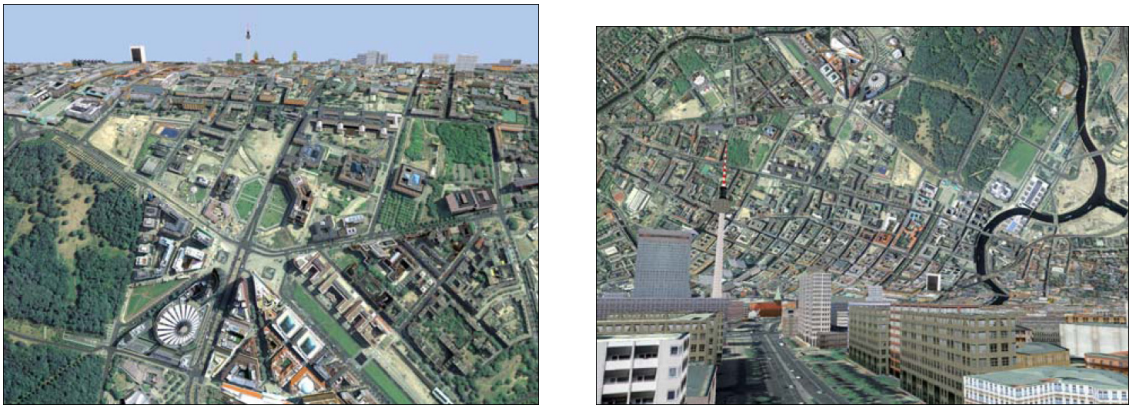


Figure 3.16: Lorenz et al. [LTJD08] use deformations on maps with 3D geometry to transition between birds-eye and horizon views (left) and between pedestrian and top-down views (right). Images licensed for use with permission [LTJD08, p. 308, 311] © Springer 2008.

horizontal cylinder. The second presents a perspective view of the street near the viewer and bends the city upward in the top of the image to appear perpendicular to the viewer in the distance. Example of both deformations are shown in Figure 3.16.

Another recent work that makes use of deformation before projection is that of Degener and Klein [DK09]. The goal of the work is to create panoramic style maps where important features are clearly shown, despite possible occlusion, in a manner that does not distort the geometry beyond recognition. In their system an initial viewpoint and direction are chosen. Optimization is used to maximize the visibility of selected features and minimize the overall deformation.

Camera-based deformations are processes similar to, but not the same as nonlinear projections. Through altering the geometry of the scene, rather than the projection these techniques for image creation utilize the following steps: (1) determine viewer position/viewing direction (2) use this knowledge to deform the scene's geometry (3) projection and lighting as normal. These steps are much easier to integrate into the standard graphics pipeline than nonlinear projections as they require no changes to projection, clipping, and interpolating operations. However these approaches use of deformation ex-

ist as independently of one another and present their own challenges in terms of modeling, controlling, and maintaining coherence in their deformation operations.

3.8 Summary

This chapter has presented and categorized a variety of nonlinear projection techniques from the current literature. In the next chapter we introduce the Flexible Projection Framework. Later, in Chapter 6 we will show how these projections can be recreated with Flexible Projection.

Chapter 4

Flexible Projection Framework

This chapter [1] describes a framework for representing projections where the projections' viewing volume is explicitly parametrically modeled. This chapter lays down the theoretical framework of Flexible Projection; the next chapter addresses how these volumes might be modelled, and Chapter 7 describes how they are rendered.

One can think of the operation of this framework as starting with an orthogonal projection's viewing volume (Figure 4.1) made from an elastic material. This viewing volume can then be deformed and manipulated to create the desired volume and projection (Figure 4.2). The end result can be a curved volume, with curved near and far clipping surfaces and curved projectors.

We represent a projection's viewing volume as the parametric volume:

$$Q(u,v,t) = \begin{bmatrix} x(u,v,t) \\ y(u,v,t) \\ z(u,v,t) \end{bmatrix}, \begin{array}{c} u_0 \le u \le u_1 \\ v_0 \le v \le v_1 \\ 0 \le t \le 1 \end{array} .$$

The parameter t corresponds to depth within the projection's viewing volume while u and v identify position within the remaining dimensions (and eventually a position in the resulting image).

To illustrate $Q(u,v,t)$ let us begin with the orthographic projection shown in Figure 4.1 and abstract it to a more general setting. For orthogonal projections, the viewing volume is a rectangular prism. From this volume, we can extract two important parametric surfaces, the near plane, $Q_n = Q(u,v,0)$, and the far plane, $Q_f = Q(u,v,1)$.

[1] The material presented in this chapter has been adapted and extended from [BSCS07] with permission, © ACM, Inc 2007.

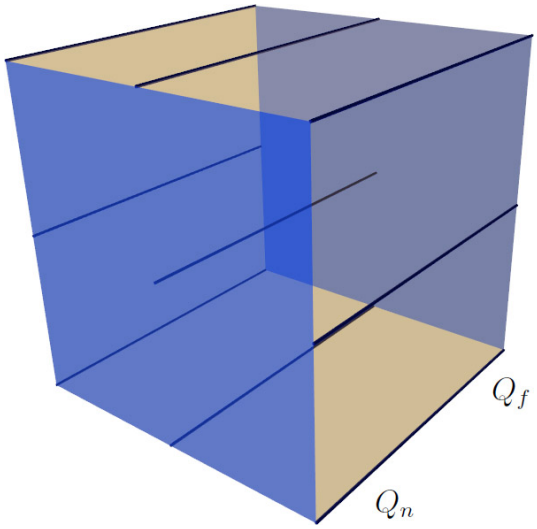


Figure 4.1: Viewing volume of an orthogonal projection. We can use this as the starting point for Flexible Projection viewing volumes that can be deformed into various shapes. Projectors are shown as dark blue lines and surfaces within the volume are transparent blue. Q_n and Q_f label the near and far planes of the volume.

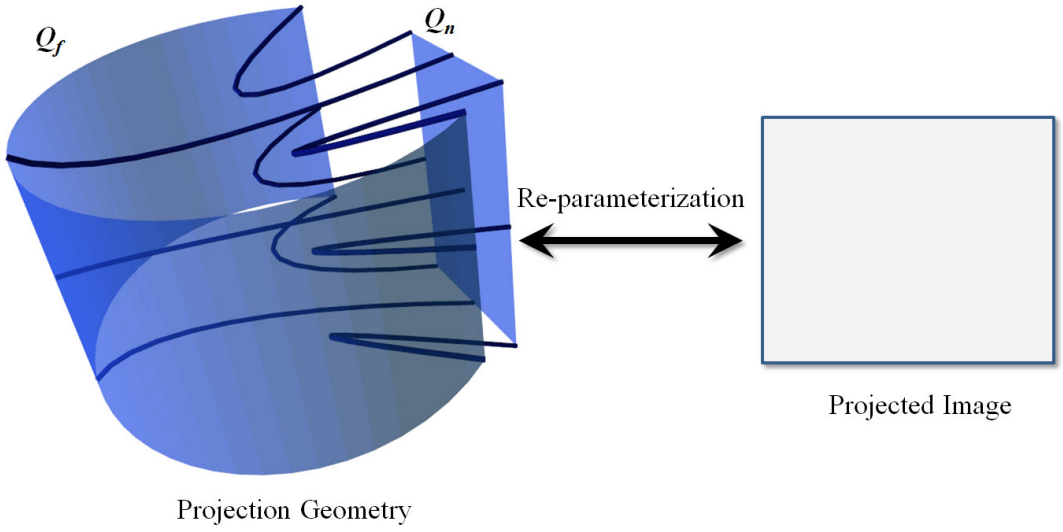


Figure 4.2: A complex projection's geometry with nonlinear projectors. Re-parameterization provides a map between Q_n and an image.

Projectors within the orthogonal viewing volume are parallel lines running between, and are orthogonal to, the near and far planes. Each projector $p_{u,v}(t) = Q(u, v, t)$ within the volume originates at $Q_n(u, v)$ and ends at $Q_f(u, v)$. If given an arbitrary point p within this volume, we determine its projection by identifying the u, v coordinates of the projector that intersects p. For more unusual volumes there may be several projectors that intersect p to identify. When creating camera-like projections (e.g., perspective, orthogonal, fish-eye) projectors are linear, composed of rays defined by two points: $Q_n(u, v)$ and $Q_f(u, v)$. We discuss projections with linear projectors further in Section 4.1.

To allow full control over all three dimensions of our parameterization, we also allow for nonlinear projectors. That is, projectors that follow curved paths through the 3D scene. The reasons for and effects of allowing this freedom are further discussed in Section 4.2. Figure 4.2 provides an example of a viewing volume with nonlinear projectors.

One remaining issue is that after we have deformed the viewing volume, our surfaces Q_n and Q_f may no longer be planar. Consequently we may no longer have a concrete viewing plane within the volume. In these cases we use an extra step, re-parameterization, to map projected points from $Q_n(u, v)$ to a viewing plane (see Figure 4.2). In most cases re-parameterization is usually accomplished by a linear mapping of (u, v) to image coordinates. Additionally depth (t) may be re-parameterizated to achieve specific depth mappings, such as the perspective depth mapping discussed in Section 2.2.3. Re-parameterization is further described in Subsection 4.3.

To render a given point $p = (x, y, z)$ within the viewing volume we must determine its parametric representation (u, v, t). To accomplish this, we identify the u, v coordinates of the projector(s) that intersect p and then determine its depth along the projector(s). This step can be expensive in the general case, however there are important cases that lead to simple computations; we will discuss this in Chapter 7.

Figure 4.3 introduces a diagrammatic representation of the viewing volume. These

diagrams present orthogonal views of the viewing volume with attention to the key surfaces, Q_n and Q_f, that can be used to identify the characteristics and behavior of the projection. In addition the blue lines provide a sampling of the viewing volume's projectors. Identifying where points in the scene will be projected can be accomplished by following the projectors through the volume. The surfaces in these diagrams are shown in green.

4.1 Projection with Linear Projectors

When working with linear projectors it is possible to define the volume entirely by specifying the two parametric surfaces Q_n and Q_f. It is important to note that these surfaces need not be rectangles or even planar. The volume covered by the projectors between these surfaces becomes the viewing volume of the projection. We represent this volume parametrically as:

$$Q(u,v,t) = (1-t)Q_n(u,v) + tQ_f(u,v) \quad t \in [0,1], \quad u \in [u_0,u_1], \quad v \in [v_0,v_1]$$

where (u_0,u_1) and (v_0,v_1) represent the intervals of the parameters used for Q_n and Q_f. We can construct projectors that begin at Q_n and lead to Q_f. A projector that originates at $Q_n(u_0,v_0)$ is defined by:

$$q_{u_0,v_0}(t) = (1-t)Q_n(u_0,v_0) + tQ_f(u_0,v_0) \quad t \in [0,1].$$

Points on a particular projector that are at a depth (t) greater than one or less than zero are not included in the image. This provides a far and near clipping distance for each projector.

It is important to show Flexible Projection is capable of exactly reproducing a perspective projection. The left image of Figure 4.3 shows perspective projection created using two square patches with normals parallel to the z-axis. Our surface Q_n is a small

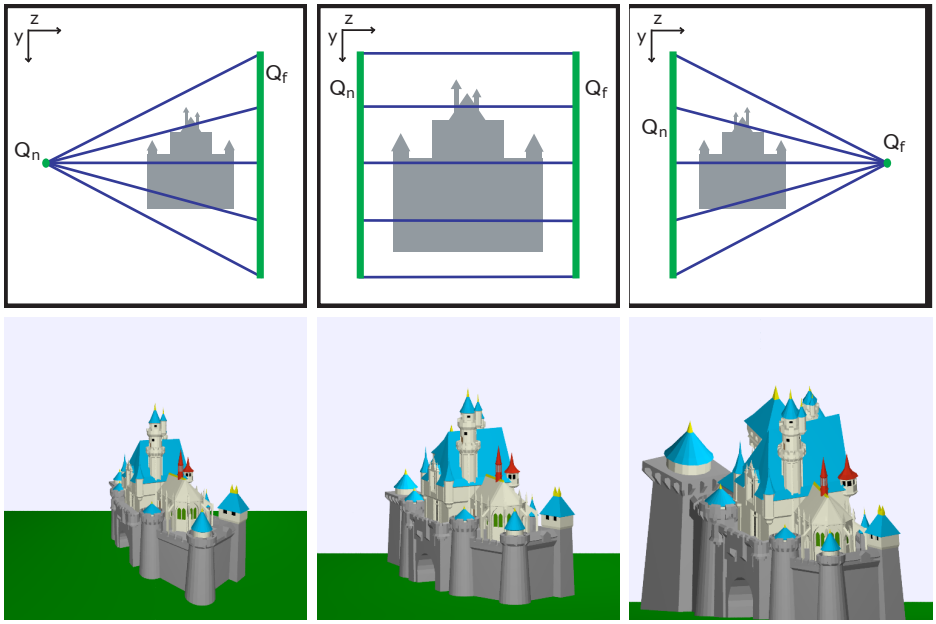


Figure 4.3: Creating perspective (left), orthogonal (center), and inverse perspective projections (right). The diagram of the setup of each projection is shown on the top while resulting image is shown on the bottom.

square and our surface Q_f is a larger square. These parametric squares are defined as:

$$\begin{bmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{bmatrix} = \begin{bmatrix} c_x + lu \\ c_y + lv \\ c_z \end{bmatrix}$$

where $c = (c_x, c_y, c_z)$ is the center of the square and l is half the width of the square. To test reproduction of perspective let us set $c_n = (0,0,1)$ and $c_f = (0,0,2)$, $l_n = 1$, and $l_f = 2$. With Q_n and Q_f defined, our viewing volume is:

$$Q(u,v,t) = (1-t)[c_n + (1,0,0)u + (0,1,0)v] + t[c_f + (2,0,0)u + (0,2,0)v].$$

This leads to

$$\begin{bmatrix} x(u,v,t) \\ y(u,v,t) \\ z(u,v,t) \end{bmatrix} = \begin{bmatrix} (1-t)u + 2tu \\ (1-t)v + 2tv \\ (1-t) + 2t \end{bmatrix}.$$

We solve for $p^* = (u, v, t)$ and find:

$$\begin{bmatrix} u \\ v \\ t \end{bmatrix} = \begin{bmatrix} \frac{x}{z} \\ \frac{y}{z} \\ z - 1 \end{bmatrix}.$$

Aside from the depth mapping we now have a perspective projection. It is straightforward to reparameterize $t = \left(\frac{fn}{n-f}\right)\frac{1}{z} - \frac{fn}{n(n-f)}$ to achieve the exact mapping discussed in Section 2.2.3. By varying the sizes of Q_n and Q_f, as well as their centers, while ensuring that the projectors converge to a single point, we can recreate other perspective projections.

Similarly for inverse perspective where Q_n is a larger square and Q_f is a smaller square with the settings $c_n = (0, 0, 1)$ and $c_f = (0, 0, 2)$, $l_n = 2$, and $l_f = 1$. The corresponding viewing volume is:

$$Q(u, v, t) = (1 - t)[c_n + (2, 0, 0)u + (0, 2, 0)v] + t[c_f + (1, 0, 0)u + (0, 1, 0)v].$$

This leads to

$$\begin{bmatrix} x(u, v, t) \\ y(u, v, t) \\ z(u, v, t) \end{bmatrix} = \begin{bmatrix} (1 - t)2u + tu \\ (1 - t)2v + tv \\ (1 - t) + 2t \end{bmatrix}.$$

As before we solve for $p^* = (u, v, t)$ and find:

$$\begin{bmatrix} u \\ v \\ t \end{bmatrix} = \begin{bmatrix} \frac{x}{3-z} \\ \frac{y}{3-z} \\ z - 1 \end{bmatrix} = \begin{bmatrix} \frac{x}{e-z} \\ \frac{y}{e-z} \\ z - 1 \end{bmatrix}$$

since e, the distance from the origin to the eye position is three. Aside from the depth mapping we now have an inverse perspective projection.

It is worth noticing that by shifting Q_n or Q_f and changing their width to height ratios in ways that cause the projectors to converge differently, we can create entirely new, but somehow related projections. Figure 4.4 shows a well known drawback of perspective

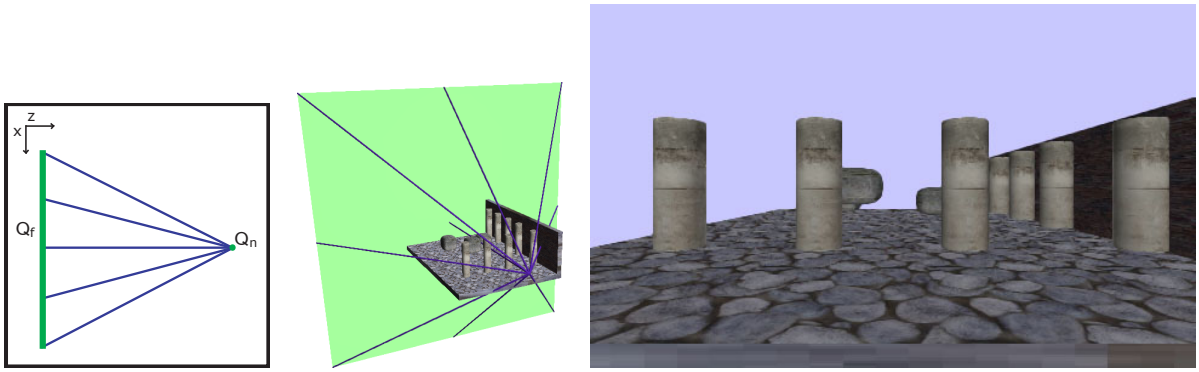


Figure 4.4: Perspective projection of a row of columns. The wide angle field of view of this projection results in the columns appearing distorted, as if they have elliptical cross-sections. Left: diagram representation of perspective projection. Center: projectors and the far clipping surface shown in relation to the 3D model. Right: the resulting image.

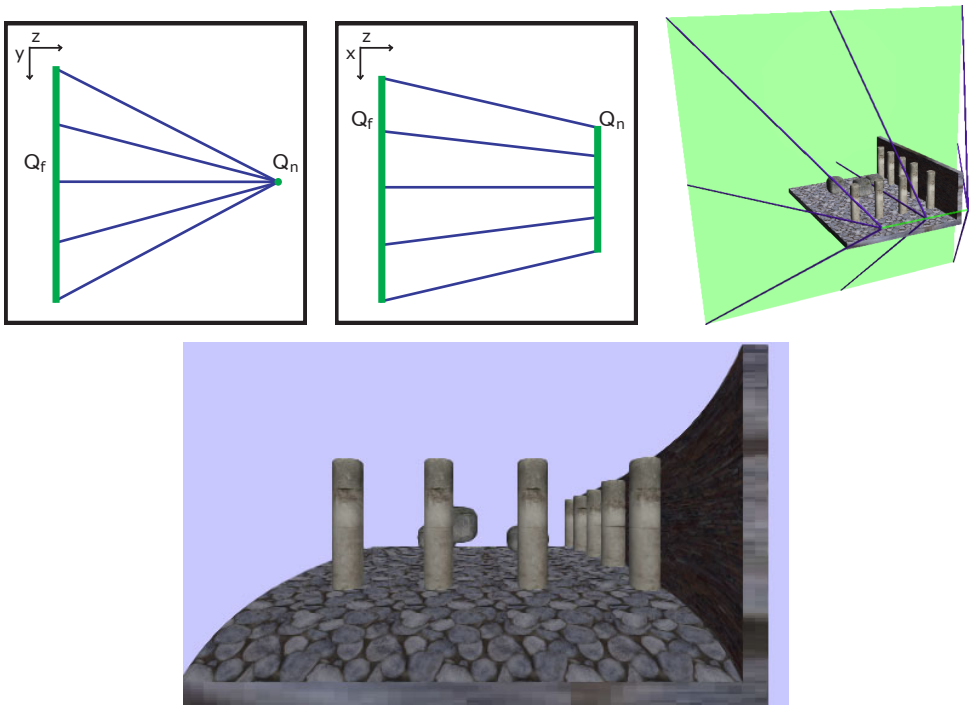


Figure 4.5: Use of an irregular perspective projection where the near surface has been altered to form a horizontal line in order to correct the distortion of the column's cross–sections shown in Figure 4.4. Top left and center: diagram representations of the irregular projection from the side and top. Top-right: projectors and near and far clipping surface shown in relation to the 3D model. Bottom: the resulting image.

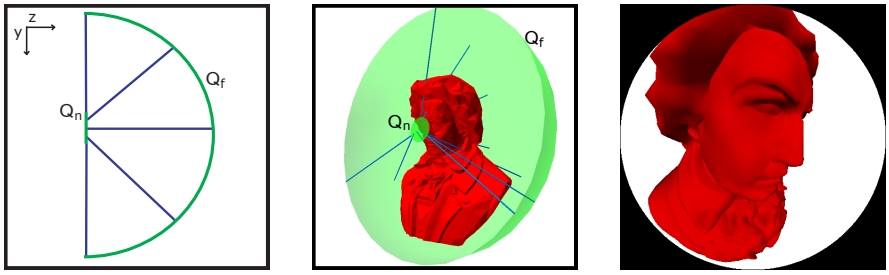


Figure 4.6: A diagram of (left) and the 3D setup of (center) an angular fish-eye projection. The right image is the result of this projection.

projection: when displaying a row of columns with a wide viewing angle the columns appear distorted, as if they have elliptical rather than circular cross-sections. Figure 4.5 presents an custom-defined projection where the near surface has been changed to a horizontal line. In the resulting image the columns appear more natural. This brief example begins to convey the flexibility and power of the framework to create, modify, and explore the space of possible projections.

4.1.1 Nonlinear Projections from Curved Surfaces

It is also possible to use linear projectors to create nonlinear projections by creating viewing volumes where Q_n or Q_f are curved surfaces. For example, by using a small circle (or small hemisphere) as Q_n and a larger hemisphere centered at Q_n as Q_f we achieve an angular fish-eye projection [Sal06] (Figure 4.6). Another popular nonlinear projection is the cylindrical panoramic projection. This is formed by placing a cylinder Q_n within another larger cylinder Q_f (Figure 4.7). We can even construct a projection that bends around corners by bending the Q_n surface (Figure 4.8).

The major advantage of this geometric technique for composing projections is that it becomes much easier to visualize, create, and adjust created projections. Rather than being forced to work directly with the underlying math and parameters in equations,

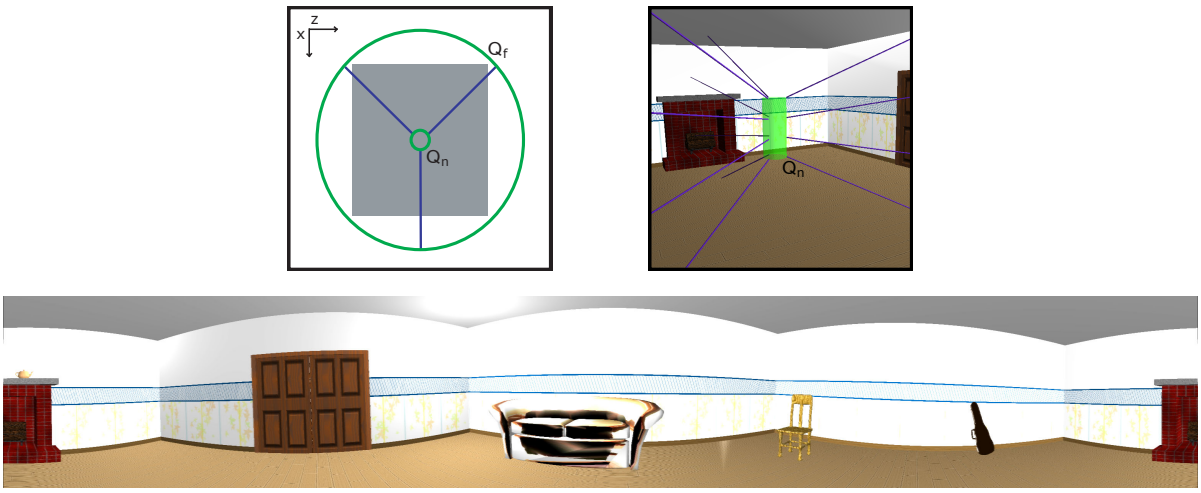


Figure 4.7: A cylindrical panoramic projection can be created with two nested cylinders.

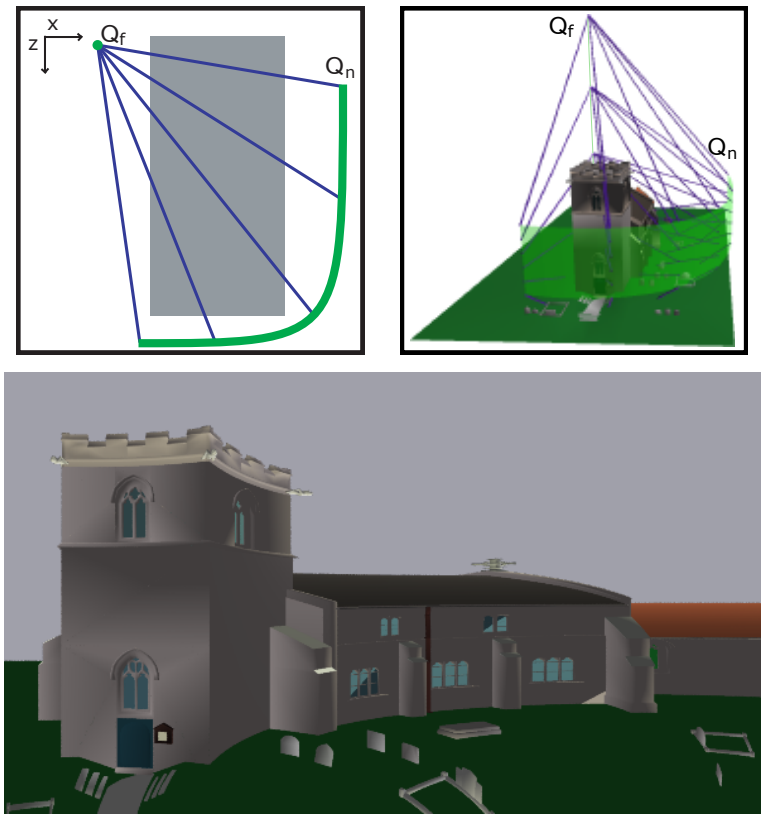


Figure 4.8: A nonlinear projection (with linear projectors) that is able to view the building from the front as well as the side.

projections are created through manipulating these relatively simple surfaces and projectors.

4.2 Projections with Nonlinear Projectors

By allowing projectors to assume curved or piecewise paths between Q_n and Q_f, we can greatly increase the diversity of projections made possible by our framework. Nonlinear projectors allow the nature of the projection to be changed, based on the depth within the viewing volume. The nonlinear projectors are parametric curves, $q_{u_0,v_0}(t)$ for any fixed (u_0, v_0).

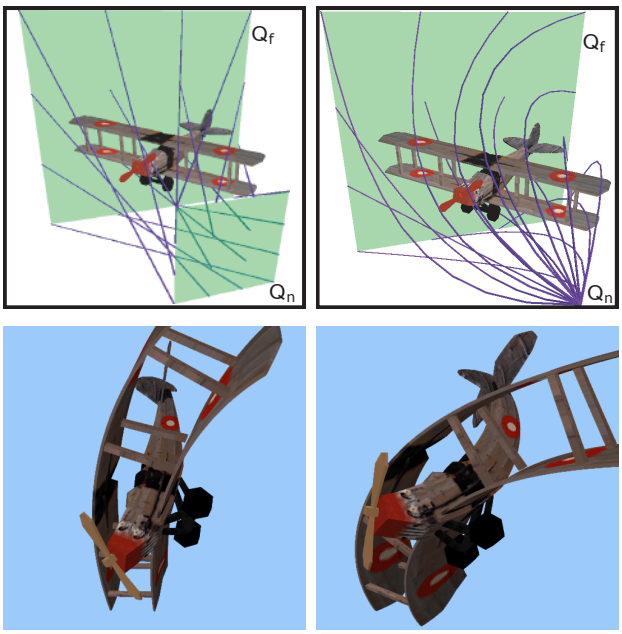


Figure 4.9: Comparison between a twist projection created with linear projectors (left) and with nonlinear projectors (right).

In Figure 4.9, we show an example of the difference in the produced image made possible by using nonlinear projectors. Nonlinear projectors allow more control over the position where a projector intersects an object, but also provide control over the angle at which the projector intersects the object. A very important aspect of nonlinear projectors

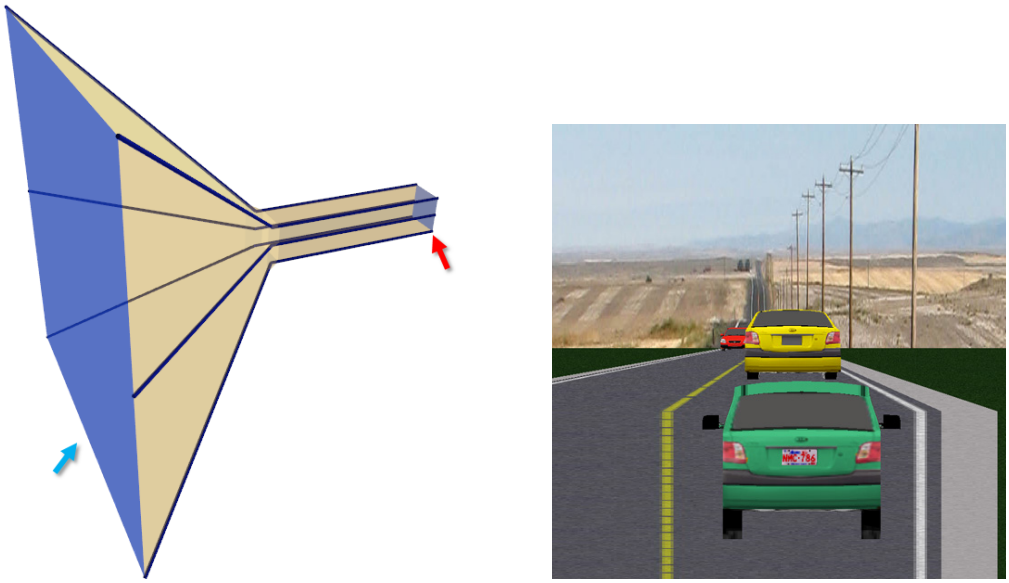


Figure 4.10: Left: The viewing volume of a projection with piecewise linear projectors that behaves like an orthographic projection close to the near surface (red arrow) and like a perspective projection as it approaches the far surface (blue arrow). Right: result of projection, notice the sharp transition between orthographic and perspective between the green and yellow cars.

is that they allow the characteristic of the volume to change through the depth of the projection. For instance, nonlinear projectors can be used to create a projection volume that acts as an orthographic projection near the camera but behaves like a perspective projection further into the viewing volume (Figure 4.10). This ability to change the behavior of the projection as a function of depth only possible in projections with curved projectors (or in multi-camera projections).

A point of interest is that in allowing projectors to follow curved paths through the projection volume we essentially are allowing light to curve. That is, viewers perceive objects in the image as if light had reflected from those objects and followed a curved path to their eyes. This raises the question of whether we should then distort shadows and other lighting calculations to maintain some sort of vector field of light through the volume. While this might provide an interesting area of exploration, my approach to this question is that such a distortion of the entire volume (and light) is better handled by a

volume deformation technique. This research concentrates on projection; that is, only the mapping of the volume to a 2D image. Consequently the 3D subject of projection, as well as this subject's interaction with light should be portrayed as it would be calculated before projection. An example of a nonlinear projection that leaves lighting and calculation unchanged is shown in Figure 4.11.

One last note is that specular highlight calculations rely on the direction of the viewer, a vector that does not have an explicit meaning with these curved projector paths. To solve this problem we use the tangent of the projector at the point of intersection between the projector and subject to provide the viewing direction. Direction of incidence for reflection and refraction calculations can be determined similarly.

4.3 Re-parameterization

With the viewing volume $Q(u, v, t)$ and a given point in the scene $p = (x, y, z)$ the projection is calculated by finding p's parameterization, $p = (u, v, t)$. Once all points have been projected, it is necessary to map (u, v, t) to an image. Within Flexible Projection this operation is known as re-parameterization. Re-parameterization can be visualized as mapping $Q_n(u, v)$ to a viewing surface (screen). Most often this viewing surface will be a plane; but, for special applications, other surfaces may prove useful. For example, if we had a curved monitor, we could map to the monitor's particular curved shape. When $Q_n(u, v)$ is a parametric surface patch, the re-parameterization is as easy as interpreting (u, v) as the width and height specifications of pixels in the image. However, if we happen to use parametric surfaces based on polar coordinates (e.g., a circle, sphere, or hemisphere) a map from polar coordinates to a rectangular shape in Cartesian coordinates can prove useful.

Aside from achieving a flat, rectangular image, re-parameterization can be used to

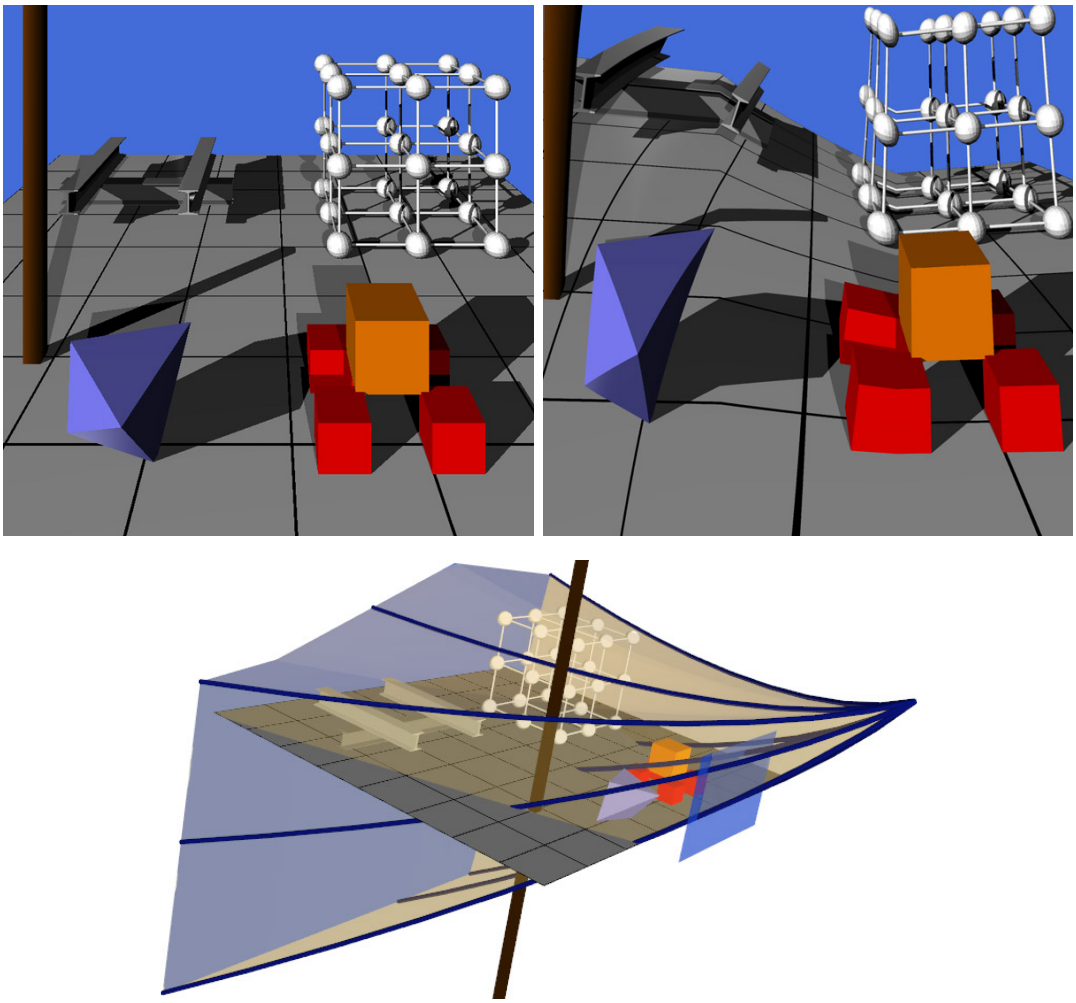


Figure 4.11: Our approach to projection leaves shadows and lighting unchanged from their presence in the original geometry. Left: a scene with shadows shown in perspective. Right: same scene after a nonlinear projection has been applied. Note that while the nonlinear projection has caused changes in the image the shadows, while stretched and skewed in places, remain at the same locations as in the perspective image. This can be seen by examining the shadows on the ground plane; although the shadows change shape with the plane, they intersect the grid lines at the same location as in the perspective image. Bottom: the viewing volume of the applied nonlinear projection.

Figure 4.12: The left two images present a diagram and the 3D setup of the angular fish-eye projection. The remaining images from center to right are the results of an angular fish-eye projection, a picture taken with a fish-eye camera lens © [Dar07] used with permission, and a re-parameterization of the angular fish-eye projection to achieve a more lens-like effect.

produce other useful effects by resampling the parameterization. A wide variety of possible distortions exist; such distortions can be taken from practically any image space distortion technique such as magic lenses [YCB05], or distortion viewing [CM01]. One possibility is magnifying a specific local area of the image to produce a magnifying lens effect. Another would be in creating a global distortion that changes an angular fish-eye projection into a lens-like fish-eye. In this distortion, shown in Figure 4.12, objects nearest the center of the circle appear larger while objects further from the center become compressed.

The re-parameterization used to map the polar coordinates of the angular fisheye's circle and hemisphere to Cartesian coordinates is

$$u = \frac{\arctan(p_y/p_x) + \pi}{2\pi}, \quad v = \sqrt{p_x^2 + p_y^2}$$

where $(u,\ v)$ are parameterized polar coordinates and p_x and p_y are Cartesian pixel coordinates that have been normalized to the range $[0, 1]$. If we compare the result produced by this parameterization (Fig. 4.12-center) to that produced by a physical fisheye lens (Fig. 4.12-second-from-right) we see that the physical lens emphasizes and stretches out the center of the image. We can adjust the parameterization of v to

$$v = \frac{1}{4}\sqrt{p_x^2 + p_y^2} + \frac{3}{4}(p_x^2 + p_y^2)$$

in order to expand the center of the image, creatinge the image shown in Figure 4.12-right. The $\frac{1}{4}$ and $\frac{3}{4}$ weights were found through experimentation of the weight that most closely reproduced the characteristics of the photograph in Figure 4.12.

In implementation with ray-casting re-parameterization includes the following steps:

1. For a given pixel (i, j) destination in the image,
2. Find (u, v) from the re-parameterization equation,
3. Form the projector $Q_{u,v}(t)$ and ray-trace it, finding the closest point p of intersection in 3D.

In order to perform scanline rendering the chosen re-parameterization must be invertible since the procedure follows the opposite order:

1. For a given triangle vertex p in 3D,
2. Find (u, v, t) from viewing volume $Q(u, v, t)$,
3. Use the re-parameterization inverse to change (u, v) into a pixel destination (i, j) in the image.

4.4 Relation to Volume Deformation

At this point Flexible Projection's parameterization of the camera's viewing volume might be seen as merely an extension of volume deformation such as free-form deformation (FFD) [SP86] or extended free-form deformation (EFFD) [Coq90]. To show otherwise, consider an analogy: the geometry of a pyramid was understood long before it was applied to create the perspective viewing system in traditional art. This viewing development, first scientifically analyzed during the Renaissance, constituted a breakthrough for viewing systems. With this in mind, the goal of this research has been to

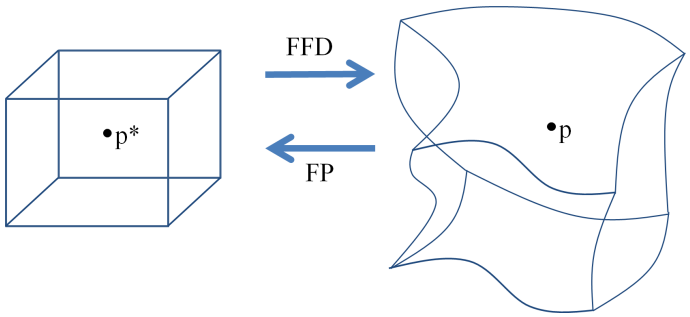


Figure 4.13: Process comparison between free form deformation (FFD) and flexible projection (FP).

present the idea of accomplishing projection by deforming the viewing volume. Notice that the goal is not to obtain (or see) deformed objects in the scene; rather it is to observe what happens to the objects after projection (in a nonlinear fashion) to a 2D surface. This creates a variety of new possibilities for creating projections.

Flexible Projection also features key technical differences from deformation. Techniques based on FFD volume deformation begin with a regular volume where correspondence between world coordinates and the 3D parameterization is easily calculated. That is, parameterization of the volume (u, v, t) is calculated from a simple volume where techniques such as linear interpolation can easily find (u, v, t) from (x, y, z). Flexible Projection is the dual of this operation, where the often difficult reverse task of determining the objects' parameterization takes place within an irregular, deformed volume, so that the object can be then transformed to appear within a regular volume (i.e., the viewbox). This comparison is shown in Figure 4.13.

Lastly, there exists a conceptual difference between Flexible Projection and deformation. Deformation changes the model, causing alterations to lighting, shadows, reflection, and refraction. Projection however is the transformation from three dimensions to two dimensions. Consequently it follows that lighting, shadows, reflections and refraction should appear as they would in the scene before projection.

Chapter 5

Specifying Projection Volumes

The purpose of this chapter is to move beyond the abstract Flexible Projection definition from the previous chapter and discuss two approaches to the creation of the projection volumes described in this thesis. A third approach, designed specifically for panoramic projections will be described in Chapter 9. However, it is important to keep in mind that these interfaces are only possible instantiations of the Flexible Projection Framework and consequently do not necessarily present the ideal possible interface(s) but rather describe possibilities that have been explored thus far.

The first interface approaches the specification of viewing volumes through user-specified control surfaces. These parametric surfaces describe the beginning and end of the viewing volume as well as how projectors pass through the volume in a very similar fashion to how control points can be used to describe a curve. These surfaces provide explicit control over the shape and how projectors traverse the interior of the viewing volume. The second interface allows a more direct connection between the image and the projection by defining individual projectors via an intersection point within the scene and the tangent at this intersection. The defined projectors are then interpolated to create the projection volume and the in-between projectors.

5.1 Control Surface Interface

The initial interface that was used in this research to specify projection volumes operates through the specification of control surfaces. This builds upon the near and far surfaces that are already used in many existing projections. For instance, perspective or

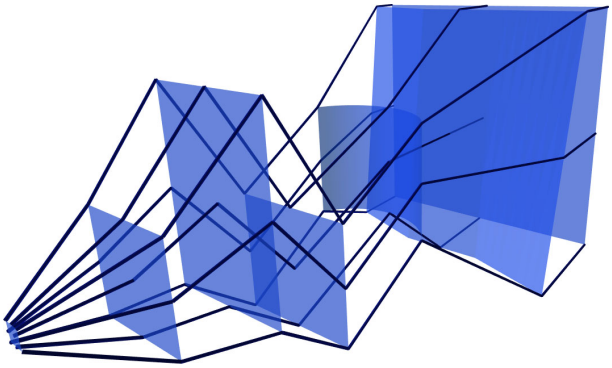


Figure 5.1: A viewing volume created from seven control surfaces.

orthographic projections are often defined by their near and far clipping planes. With this surface-based interface the parameterization of the control surface controls the image characteristics while relative placement of the surface controls how and where the projectors intercept the scene. The creation of nonlinear projectors requires more than two control surfaces.

Creating a volume in this manner is analogous to creating a curve with control points. However instead of providing control points, parameterized control surfaces are used. These control surfaces can be any sort of bivariate control surface: examples used in this thesis include: Bézier and B-Spline patches, bilinear surfaces, polar parameterized hemispheres and spheres, cylinders, bilinear patches, and more. Once the control surfaces are specified, any of a variety of techniques can be used to interpolate the volume between the surfaces. For now, as shown in Figure 5.1, assume that this is accomplished with linear interpolation. Thus for control surfaces $Q_0, Q_1, ..., Q_n$ the projection volume is defined as

$$Q(u,v,t) = \sum_{i=0}^{n} b_{i,n}(t)\ Q_i(u,v) \quad u_{min} \leq u \leq u_{max},\ v_{min} \leq v \leq v_{max},\ 0 \leq t \leq 1 \qquad (5.1)$$

where u_{min}, u_{max}, v_{min}, and v_{max} depend on parameterization of the surfaces and $b_{i,n}(t)$ provides linear interpolation between subsequent surfaces (as demonstrated in Figure 5.1).

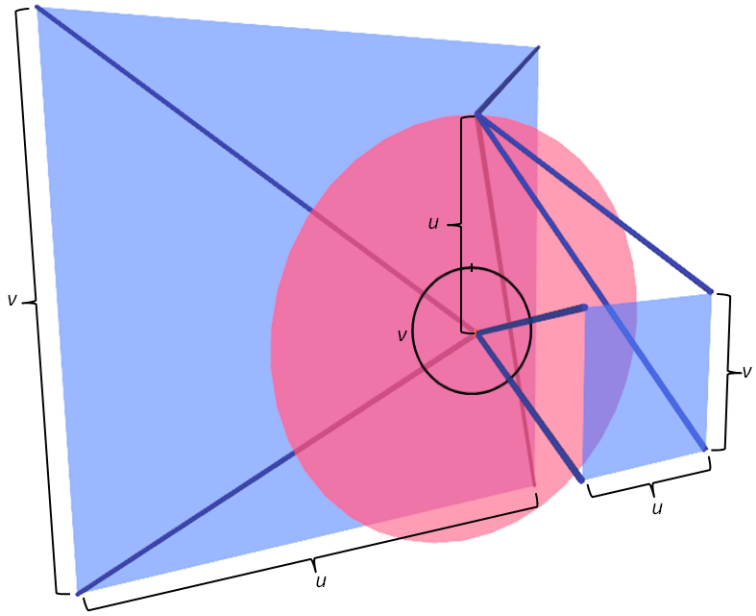


Figure 5.2: A demonstration of the importance of surface parameterization in the control surface interface. In this volume the near (Q_0) and far (Q_2) surfaces (blue) are parameterized based on Cartesian coordinates while the red, middle surface (Q_1) is parameterized by polar coordinates. The projectors have been created by linear interpolation between the surfaces. The difference in surface parameterizations causes the projectors to follow paths that are unlikely to be useful. The black brackets and lines labeled u and v indicate the extent of each surfaces' parameters' range and indicate the difference in the middle surface that has caused this problem.

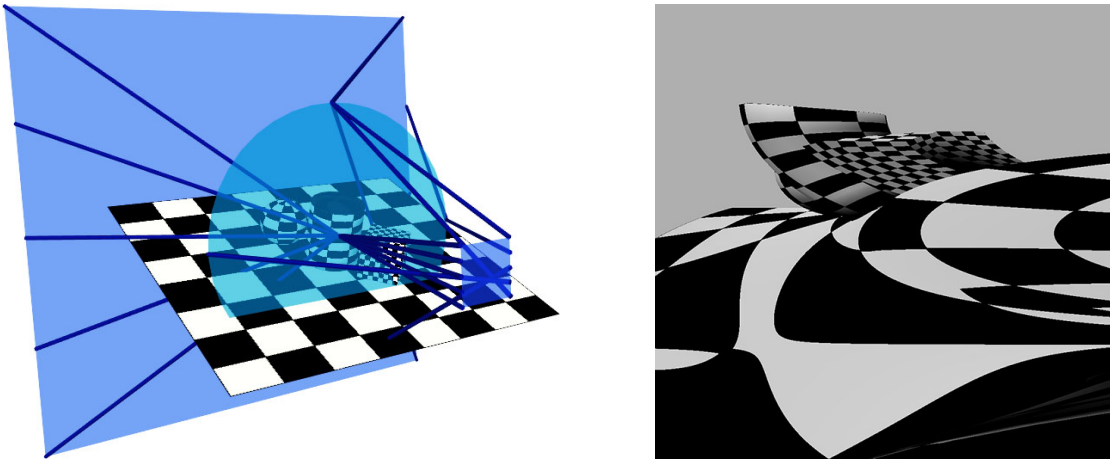


Figure 5.3: Setup (left) and result (right) of projecting a volume created from surfaces with different parameterizations (as described in Fig. 5.2). Notice that the change in parameterization causes projectors along the left edge of the near surface to all pass through the center of the circular middle surface before being redirected to the left side of the far surface. This is likely counter to the expectation that these projectors remain at the left edge of the volume.

The parameterization of the control surfaces strongly affects the characteristics of the produced image and, in most cases, should be consistently chosen for the control surfaces. The reason for this can be seen by examining an arbitrary projector $Q_{\overline{u},\overline{v}}(t)$ in a volume defined by three surfaces Q_0, Q_1, and Q_2. From Equation 5.1 this projector is the polyline defined by line segments between the points $Q_0(\overline{u},\overline{v})$, $Q_1(\overline{u},\overline{v})$, and $Q_2(\overline{u},\overline{v})$. However consider the case where Q_0 and Q_2 are parameterized based on Cartesian coordinates while Q_1 is parameterized by polar coordinates as shown in Figure 5.2. In this case $Q_{\overline{u},\overline{v}}(t)$'s neighboring projector $Q_{\overline{u},\overline{v+\delta}}(t)$ is going to move relative to $Q_{\overline{u},\overline{v}}(t)$ in one direction along Q_0 and Q_2 while moving in a different direction along Q_1. This results in very unusual and perhaps undesirable images as shown Figure 5.3. Thus to have reasonable control over the volume we have found that it is desirable to use surfaces with a similar parameterization so that, for instance, the u, v coordinates of the point at the top left corner of the first control surface corresponds to the u, v coordinates at the top left of all of the other surfaces.

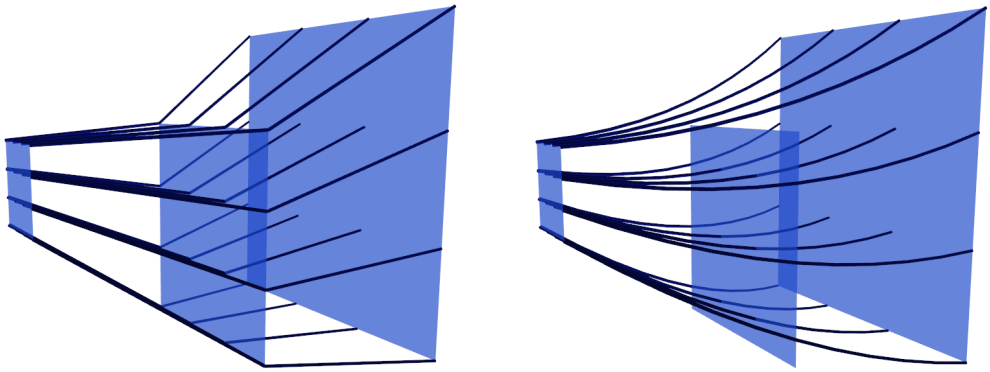


Figure 5.4: Nonlinear projectors can be created by specifying more than two control surfaces. Projectors in the left image are the result of linear interpolation while those in the right image are quadratic Bézier curves.

As discussed in Chapter 4, a wide variety of projections can be created with two control surfaces that produce volumes with linear projectors. Some common projections that can be created in this fashion include perspective, orthographic, pushbroom, spherical, and cylindrical projections. For volumes with nonlinear projectors three or more control surfaces are necessary as shown in Figure 5.4.

Linear interpolation between surfaces is one of many possibilities for "filling in" the volume. A drawback of linear interpolation is that it produces sharp transitions at each control surface. Smooth transitions can be created by using any of a variety of parametric curve basis functions. In this work Bézier curves have been used since they have the useful property of exactly interpolating the first and last control surface, which ensures that these surfaces exactly bound the volume, the same behavior as near and far clipping surfaces.

5.2 Projector-Based Interface

The goal behind the projector-based interface was to create a less abstract link between the 2D image and the 3D scene. This is achieved by allowing users to pick the projector associated with a point from image space and then specify where and from what direction

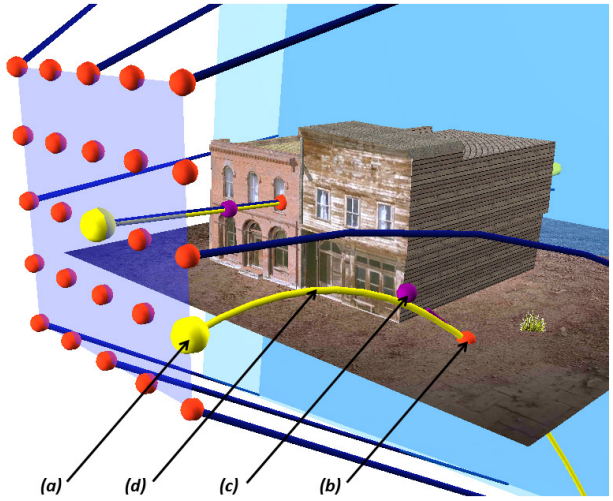


Figure 5.5: Specifying a projector in the projector-based interface. After selecting the projector's starting point on the image plane (a), a point of intersection with the scene is specified (b). The purple sphere (c) can then be moved to adjust the projector's tangent at the point of intersection. The resulting projector, shown in yellow (d), is generated by the system to meet these constraints.

this projector should intersect the scene, as shown in Figure 5.5. The viewing volume $Q(u, v, t)$ in this interface is created by interpolating between user-defined projectors.

This interface allows the direction of projectors to be easily changed and exactly specified across the viewing volume. As opposed to the control surface interface this new approach makes it easier to cause different areas of the image to behave as if projected differently since the individual projectors of that area can be directly specified. It also provides a means of recreating existing projections. That is, given an nonlinearly-projected image and a roughly equivalent 3D scene, it is possible to recreate the nonlinear projection and use it for other scenes. This will be demonstrated in Chapter 8, Section 8.3.

In this interface the users is presented with an image plane and presented with a 5×5 grid of points that indicate projectors that can be specified. This plane provides a manifestation of the image within the scene, providing a link between each projector

and the resulting image. Once a projector starting point p_n is chosen, a sphere follows the user's mouse at the 3D point where a vector originating eye and passing through the mouse's screen position intersects the 3D scene; the user clicks to fix this point of intersection p_i. The creates a straight projector that runs from p_n to p_i. The system presents another sphere connected to p_i with a line that can be rotated around p_i to control the direction v from which the projector insects p_i, allowing the projector to be curved.

A design consideration in the creation of this interface was how to interpolate between and extrapolate from the a few specified projectors to create the entire viewing volume. In general this is a difficult problem to handle automatically without strong assumptions about the desired type of viewing volume. Rather than restrict the interface to specific kinds of projections the approach taken is to instead provide a good starting point and rely upon the user to specify deviations from the starting point. Since perspective is the most commonly used projection, the system begins by configuring the projectors to create a perspective frustum. That is, when the first projector is specified the system defines the remaining twenty-four projectors ($5 \times 5 - 1$) to define a perspective viewing frustum with the specified projector as the center of projection. The user is then free to override these automatic specifications with their own. This is also the reason behind the chosen grid dimensions (5×5) of the grid: these dimensions aim to provide a balance between allowing sufficient control over the volume and having to specify a tedious number of projectors.

An implementation question is how to construct the parametric volume $Q(u, v, t)$ from the 5×5 grid of projectors. In general this can be accomplished by treating the projectors as control points in a parametric surface (e.g., Bézier, B-Spline, or NURBS patch). Our implementation provides two options for this interpolation. The first uses a bilinear patch that interpolates between each square of 4 projectors. This creates a

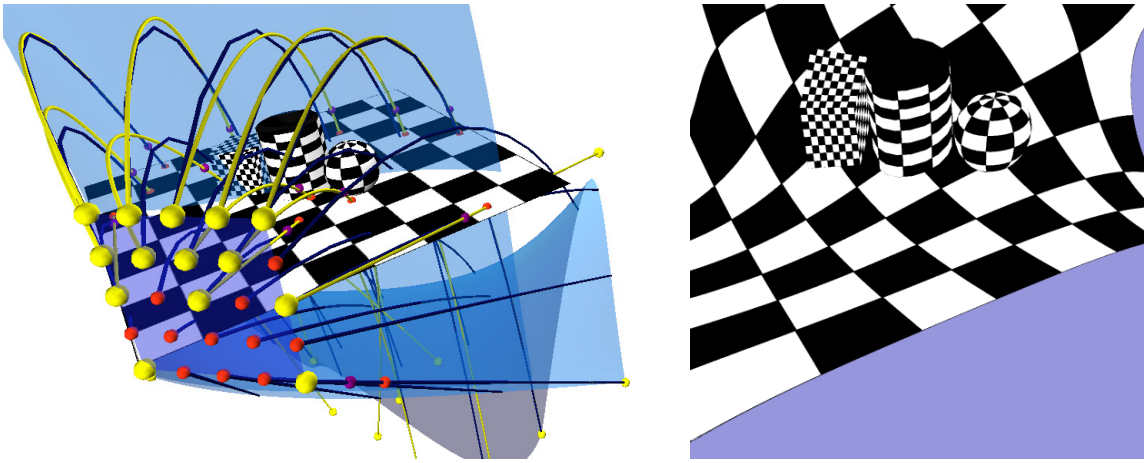


Figure 5.6: Creation of a nonlinear projection with the projector-based interface. Left: a projector-based viewing volume. The 14 specified projectors are shown as yellow curves with yellow end points. Specified points of intersection with the scene are shown as red sphere while purple spheres are used to indicate the tangent at the point of intersection. The blue curves are the projectors interpolated by the volume using Bézier based interpolation. Right: the image resulting from projection of the viewing volume.

volume where the 5×5 grid of specified projectors is exactly interpolated; all in-between projectors in each grid cell are defined by bilinear interpolation of the defined corner projectors. The second option uses a fourth order Bézier patch. This results in only the projectors at the four corners of volume being exactly interpolated, while all other projectors are only approximated. An example of this is shown in Figure 5.6

These two techniques of interpolation are compared in Figure 5.7. In this figure two projectors were user-specified. The first specified was the center projector (the upper of the two yellow lines), at this point the remaining 24 projectors were automatically created based on the assumption that they should be equally spaced over a perspective viewing frustum. The second projector's specification (the lower of the two yellow lines) specifies the projection creator's desire for that part of the image to deviate from perspective. In comparing the two interpolation approaches we can see that the volume provided by the bilinear patches (center image of Fig. 5.7) creates a strong and sharp deviation from perspective along the bottom of the car. In contrast the global support inherent in a

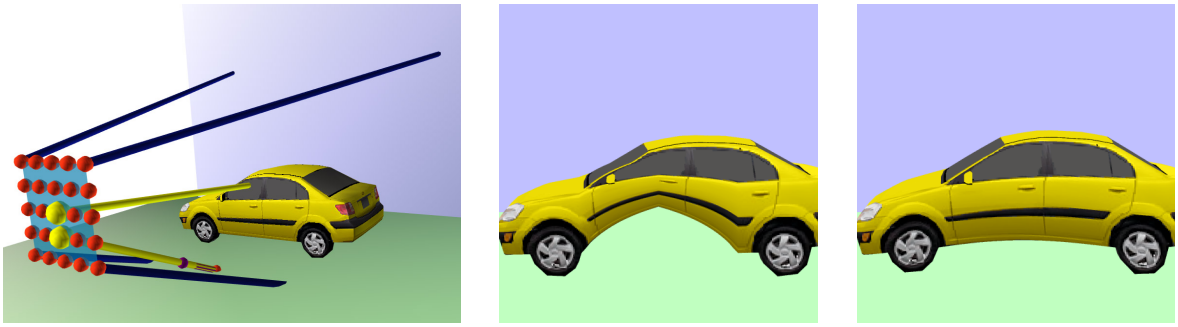


Figure 5.7: Comparison between the use of bilinear interpolation (center) and Bézier interpolation (right) of the viewing volume in the projector-based interface. The left image shows the construction of a viewing volume with the projector-based interface where two projectors (yellow) have been specified.

5×5 Bézier patch (right image) creates a subtle but smoothly transitioned curve in the car.

5.2.1 Projector Construction

As mentioned earlier, projectors are defined by selecting the position where the projector intersects the near plane p_n, the position the projector insects the scene p_i, and optionally suggesting a tangent v at the point of intersection. The projector is constructed by creating a curve that meets these conditions. In order to make rendering easier it is preferable for the curve to be linear or quadratic (see Chapter 7). Lastly, it is assumed that the direction of the tangent at the point of intersection is important but its magnitude is not, a reasonable assumption given that lighting calculations make use of a normalized tangent when determining diffuse and specular lighting contributions [Wat00].

If the tangent is not specified the system assumes that $\overline{v} = p_i - p_n$ and the solution is simple: a linear curve (i.e., a line segment) where $p_0 = p_n$ and the second point p_1 exists at some point along the vector $p_i - p_n$. To ensure that this projector as well as neighbouring projectors created through interpolation intersect the scene, we set the

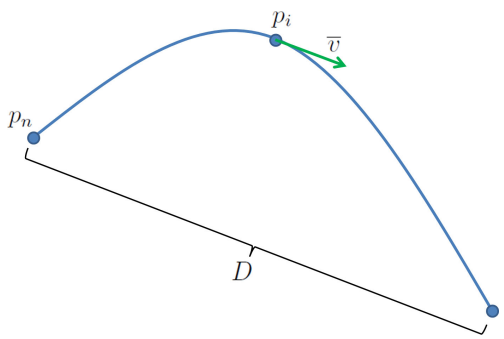


Figure 5.8: Projector construction. Given a point on the near surface p_n and a point of intersection with the scene p_i and the projector tangent at this point $\overline{v}$ (green) we need to find the Bézier control point necessary to describe the projector (blue).

projector's length to some value D that is large enough to ensure that the projector is longer than the scene. Thus our projector is $p_n + t(p_i - p_n), 0 \leq t \leq \frac{D}{||(p_i - p_n)||}$.

If a tangent is specified (as in Figure 5.8) then we must construct a curve that pass through p_n and p_i as well as have a tangent in the direction of v at p_i. Additionally the end of the projector should be a distance of D from p_n for the same reason as the linear case.

5.3 Summary

This chapter has described two possible interfaces for Flexible Projection. The first makes use of control surfaces to define the volume, much like control points are used to create curves. This interface is useful for individuals who are already familiar with projections and possess the knowledge and tools to work directly with parametric surfaces. The projector-based interface provides a more explicit link between the image and the projection volume by allowing users to specify how projectors originating from the image plane trace through and intersect the scene.

Both interfaces were designed to be quite general, supporting a wide range of different projection types and characteristics. However, these powerful tools require a great deal

of training and knowledge for users to make use of them effectively. This complexity is a natural consequence of supporting this wide variety of possibilities. Alternatively, less demanding interfaces are certainly possible for designing a limited range of projections. Chapter 9 will provide an example of this for the more constrained task of creating panoramas.

Chapter 6

Comparing Flexible Projection to Other Techniques

This chapter examines the usefulness of Flexible Projection (FP) as a framework. To ascertain its usefulness I examine its ability to reproduce existing nonlinear techniques as well as its abilities in exploring new projections that have previously been unexplored.

Section 6.1 examines FP in regard to the related work. Multi-camera projection, albeit sometimes reproducible with FP, provides the strongest contrast with this work. An in-depth comparison of these two approaches is explored in Section 6.2. The chapter is concluded with Section 6.3, where FP's ability to explore beyond existing works is discussed.

6.1 Comparison to Related Work

The Flexible Projection Framework is able to reproduce a wide variety of linear and nonlinear projections, including the majority of the nonlinear projections described in Chapter 3. The ability to reproduce existing techniques is useful in several ways. The first is that using FP as a common basis of representation allows the rendering, interface, and visualization techniques developed for FP to be used for these other projections. Additionally this reproduction provides insight on how other nonlinear projections might be combined or interpolated. Examples of combining projections have already been shown in Figure 4.5 where a projection that combines perspective and pushbroom projections is described, and in Figure 4.10 where a depth-wise combination of orthographic and perspective is described. In these scenarios FP representation provides a means of interpolation between different projections, as discussed further in Section 6.3. Lastly, if

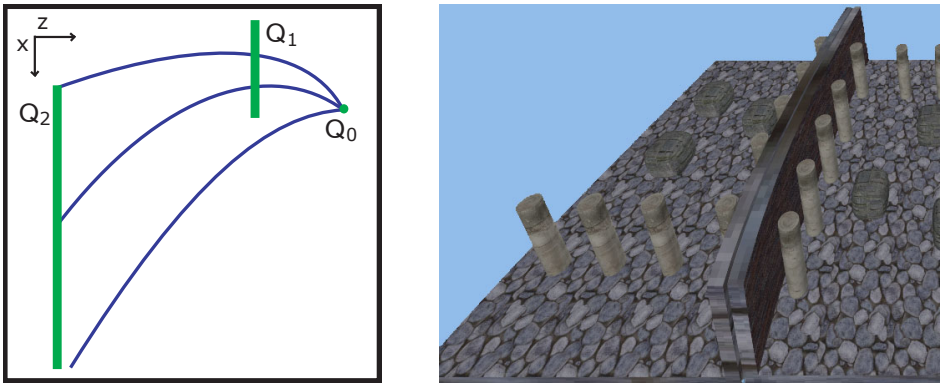


Figure 6.1: A projection that uses nonlinear projectors to create curved convergence of parallel lines. A perspective projection image of the scene is shown in Figure 4.5.

we consider the space of all possible projections, the description of FP encompasses a large subset of this space; this is in contrast to many existing projection techniques that describe a single type of projection representing an individual point in this space.

In the following discussion I describe how to re-create the geometry of other nonlinear projections with FPs. In general this description involves defining the viewing volume and then developing a parameterization of this volume. Re-parameterization can then be used, if needed, to exactly match the resulting image space. Related work has been grouped as it was in Chapter 3.

6.1.1 Curved Projection Surfaces

In regards to Inakage's nonlinear projections [Ina91] and Levene's framework [Lev98], Figure 4.3 has already demonstrated FP's ability to control convergence and divergence of parallel lines within the scene. A demonstration that FP can also produce curved convergence of parallel lines is shown in Figure 6.1.

Optical Models [WM90], Digital Cubism [Gla00], Distortion Cameras [AG95], and Single-Center Projections [TD08] all define nonlinear projections through independent specification of the projectors' origins and direction. For example in digital cubism projectors are defined by two NURBS surfaces, one providing the projector origin and

the other providing direction. This can be produced in FP through linear interpolation between the same two NURBS surfaces. The far clipping surface of the volume can be defined by enforcing a range on t the variable used to interpolate between the two surfaces. The reproduction of the other techniques (Optical Models, Distortion Cameras, and Single-Center Projections) is similar. Since all of these techniques only consider projections with linear projectors they can be considered as representing a subset of the projections possible with FP.

Single-Center Projections [TD08] introduce the possibility of creating discontinuous viewing volumes, as was shown in Figure 3.3. A discontinuous viewing volume can be created in FP through use of discontinuous parametric surfaces in the control surface interface, a diagram of such as shown in Figure 6.2, an actual recreation with FP is shown in Figure 6.3.

6.1.2 Cameras

As several nonlinear camera projections (i.e., Pushroom [GH97], Crossed-Slit [ZFPW03]) can be reproduced with General Linear Cameras (GLCs) [YM04b] it is important to show that FP can reproduce any GLC.

A GLC is defined by three rays that intersect the image plane at $z = 0$ and another plane at $z = 1$. The rays pass through $z = 0$ at $(u_i, v_i, 0)$ for $i = [1, 2, 3]$ and the $z = 1$ at $(s_i, t_i, 1)$. Consequently, each ray can be parameterized in 4D by (s_i, t_i, u_i, v_i). A GLC produces an image by collecting measurements from the affine combinations of the rays:

$$r = a(s_1, t_1, u_1, v_1) + b(s_2, t_2, u_2, v_2) + (1 - a - b)(s_3, t_3, u_3, v_3).$$

Since a and b are essentially barycentric coordinates that simultaneously parameterize both the $z = 0$ and $z = 1$ planes, we can use a and b as the parameterization of the FP surfaces Q_n and Q_f that exist at depths $z = n$ and $z = f$ as shown in Figure 6.4. This

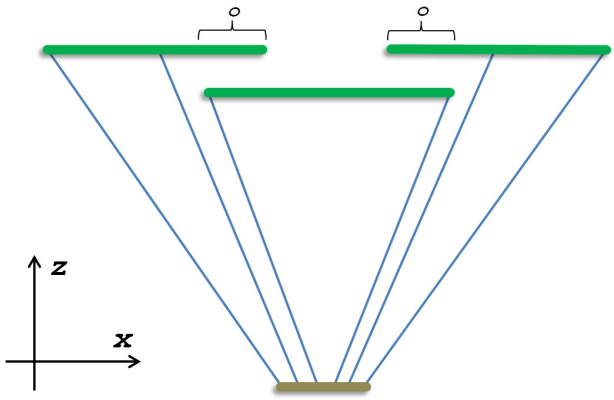


Figure 6.2: A top-down diagram of discontinuous projection volume. In this volume the far surface (green) is discontinuous. The projectors (blue) in the lower portion of the far surface span a larger region of the scene. Additionally some parts of the scene may be represented more than once over the regions marked *o*.

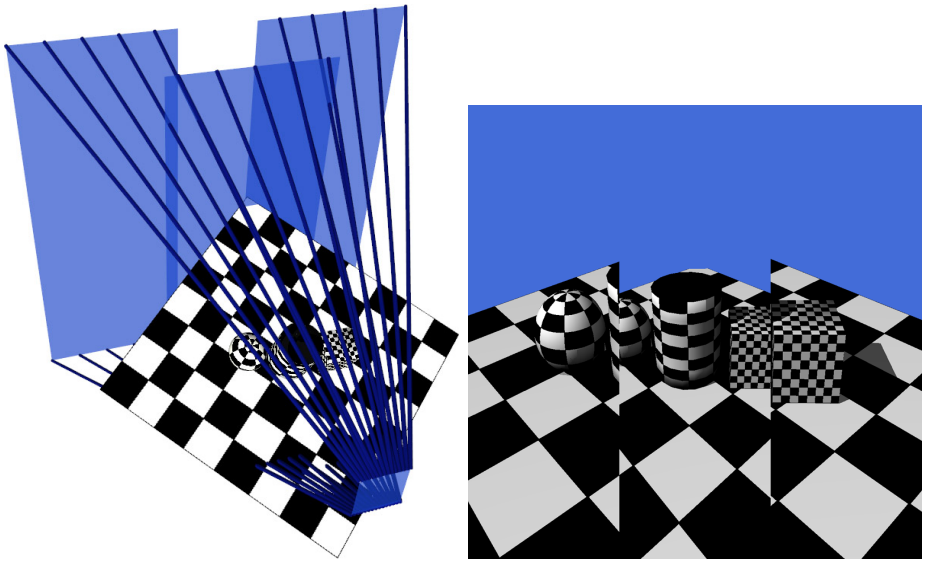


Figure 6.3: A discontinuous projection created with Flexible Projection

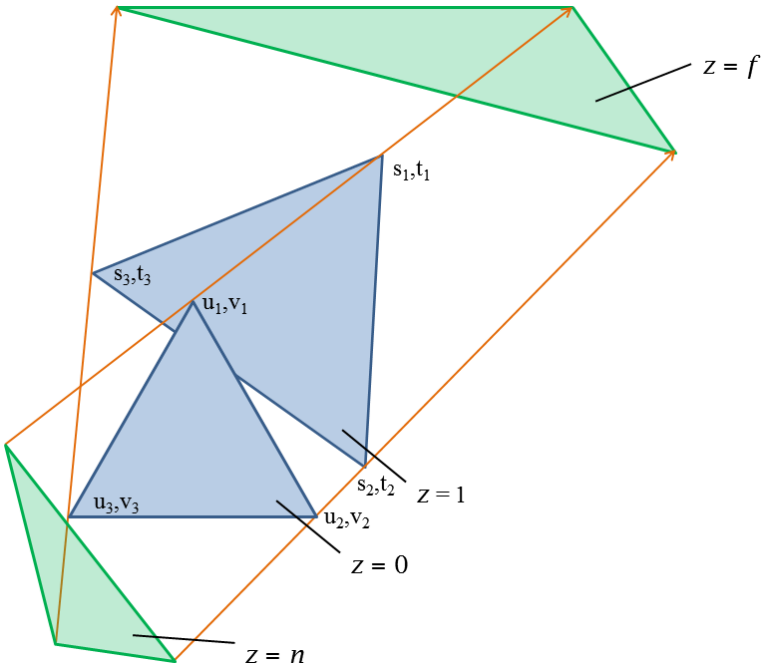


Figure 6.4: Creating a Flexible Projection from a General Linear Camera (GLC). The triangles that are used to define the GLC are shown in blue. The triangle that are used in the corresponding Flexible Projection, shown in green, are created by extending the projectors at the corners of the GLC triangles.

creates the FP volume $Q(a, b, t)$. Q_n and Q_f become the planes that bound this volume and, since rays are linear, we use a linear term for t as:

$$Q(a, b, t) = (1 - t)Q_n(a, b) + tQ_f(a, b).$$

An additional point of comparison is that GLCs cannot produce nonlinear projectors as any affine combination of linear rays (the rays defining a GLC) produces linear rays. Throughout the thesis we have already demonstrated Flexible Projection's ability to recreate several GLC projections including perspective, parallel, inverse (all shown in Fig. 4.3) and pushbroom (Fig. 2.11) projections.

Rademacher and Bishop's [RB98] multiple-center-of-projection (MCOP) images can be reproduced using an FP with linear projectors. As shown in Figure 6.5 the surface Q_n takes the form of an arc-length parameterized curve that follows the MCOP path of the camera, while Q_f is defined in a piecewise fashion based on the far plane resulting from the MCOP camera parameters. As described by Rademacher and Bishop these parameters are (C_i, O_i, U_i, V_i) for each column i of the MCOP image where C is perspective eye position, O is a vector from C to the image plane origin, and U and V respectively define the horizontal and vertical axis of the projection planes.

The Rational Lens Distortion Model [CF05] analyzes images from unknown, uncalibrated cameras and calculates a mapping to a perspective image. Such a mapping can be used in the re-parameterization step of Flexible Projection.

RTCams [HCS+07], as discussed in Section 3.2 are defined by third-order tensors. Describing a projection volume for RTCams in general is like attempting to describe a general projection volume for a matrix; difficult and complex in general but possible for specific instances (e.g., perspective). Thus while more elegant solutions may exist for specific classes of RTCam tensors, the following description provides a method for constructing an FP viewing volume for RTCams in general. Hall et al.'s [HCS+07]

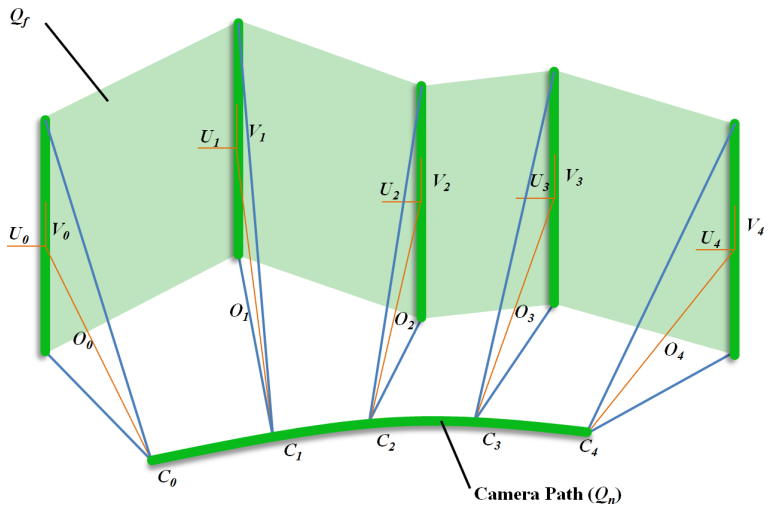


Figure 6.5: Creating a viewing volume for a MCOP image. This image demonstrates the construction of Q_n from the camera path while Q_f is constructed from piecewise interpolation of the far planes resulting from the individual perspective projections that make up the MCOP image. Each individual projection's parameters C, O, U, and V are shown with orange lines.

definition of RTCams includes a process for extracting projectors (which are referred to as rays in [HCS+07]). This process [HCS+07, p. 970] involves, for a point on the image y, finding a matrix of partial derivatives for a given 3D point p on the projector and then using this matrix to find the direction such that $\delta y = 0$, leaving the projector stationary; integrating over all of these points yields an entire projector. This process can then be repeated for as many projectors in the volume as needed to accurately reproduce the volume.

The Occlusion Camera [MPS05], designed to capture occluded regions of a scene, can also be reproduced with FP. The FP reproduction's viewing volume uses piecewise linear projectors defined by three control surfaces. The first, Q_0 is a point at the camera position. The second, Q_1 is the plane that Mei et al. refer to as the near distortion plane. The third surface uses the exact parameterization described in their paper; that is $Q_3(u, v) = (u, v) + \frac{(u-u_0, v-v_0)}{||(u-u_0, v-v_0)||}$ where (u_0, v_0) is the specified pole parameter of the

occlusion camera (usually the coordinates of the centroid of the occluder) [MPS05, p. 339].

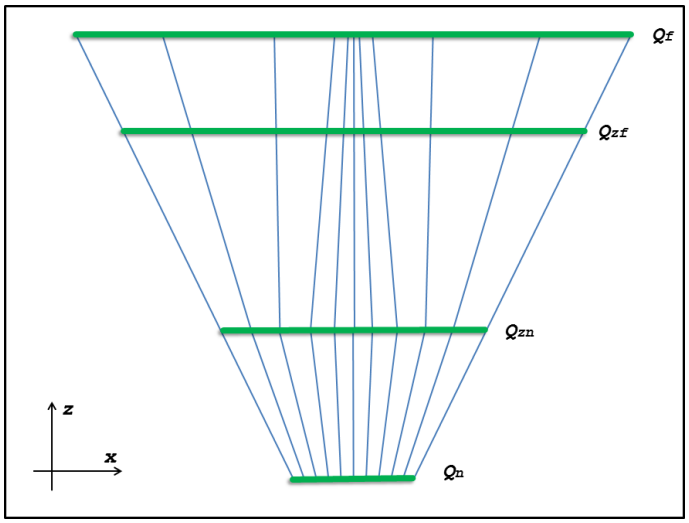


Figure 6.6: Reproducing a depth discontinuity occlusion camera with a Flexible Projection. The diagram portrays a top-down slice of the viewing volume where the depth discontinuity is present in the middle of the shown volume, positioned over an area at the center of Q_{zn}.

Depth Discontinuity Occlusion Cameras (DDOCs) [PA06] can also be reproduced using piecewise linear projectors. To construct a FP viewing volume for this type of projection one begins with a perspective viewing volume with two additional control surfaces Q_{zn} and Q_{zf} interpolated between the near and far planes (Q_n and Q_f). The relative depth of Q_{zn} and Q_{zf} between Q_n and Q_f is dependent Z_n and Z_f parameters of the DDOC. The presence of Q_{zn} will assure that the projection between Q_n and Q_{zn} produces exactly the same image as a perspective projection. The DDOC distortion is accomplished by adjusting the parameterization of Q_{zf} and Q_f using the same distortion map that defines DDOCs. This will cause the distortion to be introduced between Q_{zn} and Q_{zf} and maintained thereafter as shown in Figure 6.6.

6.1.3 Lenses

As previously described, lenses create images through localized distortions, usually in image space. As such the results obtained by these techniques can often be represented by a 2D to 2D mapping. Consequently adjusting the shape of the Flexible Projection's viewing volume is not generally necessary. Instead we should adjust the distribution of the projectors sampling the volume. This is most easily achieved by the re-parameterization operation describe in Section 4.3.

Since Magic Lenses [YCB05] are created by a strictly 2D to 2D mapping, magic lenses can be incorporated into any Flexible Projection by using this mapping as the re-parameterization operation. The Elastic Presentation Framework [CM01] lenses can be reproduced in a similar fashion by constructing the same perspective projection as used in the elastic framework and then incorporating the elastic mapping as a re-parameterization operation. The use of both of the lenses' 2D mappings as re-parameterizations highlights the potential use of lens-based techniques as a useful and already existing interface for controlling the re-parameterization operation used by Flexible Projection.

The effects achieved with Carpendale et al.'s [CCF97] 3D occlusion lenses where a select focus point within a 3D volume remains unobscured or enlarged due to deformation of the volume's space are likely most easily accomplished with deformation, however they can also be produced with Flexible Projection. For example, increasing the size of a selected point in the volume can be achieved by decreasing the space between projectors in that region of the volume. The occlusion effect, namely moving objects that obscure a selected region by displacing away from the vector between the region and the eye position can alternatively be accomplished through the use of nonlinear projectors. Projectors can be assigned curves that bypass obscuring objects to focus on the desired region. An in-depth example by using nonlinear projectors to bypass obscuring objects is described as an application of Flexible Projection in Section 8.1.2.

In Magic Volumes [WZMK05], three different projections that make use of linear projectors are described. All three projections begin as orthographic projections. The first, called the magnification lens, changes the direction of a subset of rays to point toward a point on the far plane, instead of remaining parallel. This single point is surrounded by a transition region where the ray direction is pointed slightly toward this single, magnification point. This effect can be produced with Flexible Projection's control surface interface using high-degree, planar B-spline patch with a fine resolution of control points. The magnification point can be created by placing several neighboring control points at a single position and by moving surrounding control points closer to this location. The second projection, the sample rate-based lens, is produced by changing the sampling of the image plane to produce areas of magnification. This is easily accomplished with Flexible Projection by using an appropriate re-parameterization mapping. Fisheye lenses are the third projection presented for Magic Volumes; an example fisheye produced with FP has been shown in Figure 4.12.

6.1.4 Cartography

Much like lenses, cartographic projections exist as 2D to 2D mappings, although in this case one of the surfaces is the surface of a spheroid rather than a plane. An equidirectional spherical projection as described by [Sal06] creates a rectangular image where lines of latitude and longitude form straight lines on the image. This projection is essentially the simplest cartographic projection from which many other cartographic projection are described as 2D to 2D mappings. Thus, if we can use FP to create an equidirectional spherical projection then we can also use these mappings as the re-parameterization step of Flexible Projection and incorporate all such cartographic projections into the framework. An equidirectional spherical projection can be created within FP where Q_n

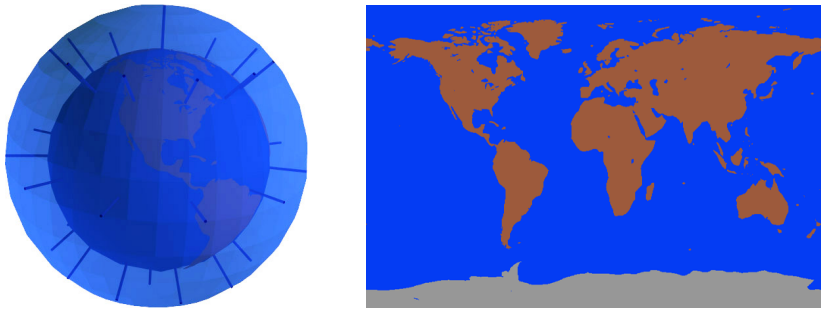


Figure 6.7: An equidirectional spherical projection created from nested spheres (Q_n is the outside sphere).

and Q_f are two nested spheres (or earth-shaped ellipsoids if high accuracy is required) that have been parameterized with polar coordinates as demonstrated in Figure 6.7.

6.1.5 Implicitly Defined Projections

Implicitly defined projections rely on an implicit definition of the projection volume, usually through a vector field defined by a dynamic system. The challenge in representing such projections as FPs is to extract explicit projector definitions from the vector field. Nonlinear ray-traces are the usual mechanism for rendering these projections, as such we can use the nonlinear ray tracer's initial starting locations as a near surface for the FP viewing volume. From this near surface of points, we can use Gröller's technique [Gr5] that extracts piecewise rays (projectors) from the vector field. If we do not wish to integrate every projector, we can extract a grid of projectors through the volume and then interpolate between these known projectors. The number of projectors required to form such a grid and produce an accurate interpolation will depend upon the vector field's properties but likely will be significantly less than the resolution of the desired projected image. It is important to note that no other projection framework encapsulates nonlinear ray tracers.

6.1.6 Multi-Camera Projection

Before discussing specific multi-camera (MC) projections, it is important to note that the general approach of MC projection is significantly different than that of FP. The approach in MC is to divide the desired image into regions and then to find different linear projections for each region and attempt to blend these results into a single image. In contrast, FP attempts to model the projection as a whole, creating a single viewing volume that defines the entire projection. Consequently, there exist significant differences in the capabilities, strengths, and weaknesses of these two approaches.

From the earlier description of multi-camera (MC) projections it is clear that there exists wide variety in how MC projections are created, defined, and used. To help determine how FP can be used to reproduce MCs it is useful to separate MC projections into two categories: those that create a continuous volume and those that create discontinuous volumes.

Continuous Multi-Camera Projections

The key behavior that differentiates continuous volume multi-camera projections (CVMCPs) from discontinuous ones is that continuous MC projections preserve local neighbourhoods. That is, neighbouring parts of the image space correspond to neighbouring parts of the viewing volume; there are never regions of the image space that contain the results of two different projection volumes. Examples of CVMCPs include: photographs stitched together to form panoramas, long scenes [AAC+06], and multiperspective images with general linear cameras [YM04a].

In general this type of projection is well suited for reproduction with Flexible Projection. In order to do so, one must find the viewing volume of the multi-camera projection, determine the equivalent parameterization, and lastly describe any re-parameterization necessary when moving to image space.

Figure 6.8: A long scene multi-camera projection created by stitching together images. Image licensed for use with permission [AAC+06] © ACM, Inc 2006.

Figure 6.9: Enlargement of an intersection from the image shown in Figure 6.8. Image licensed for use with permission [AAC+06] © ACM, Inc 2006.

Stitched together photographic projections can be reproduced by cylindrical projections (as shown in Figure 4.7) or spherical projections depending on the constructed type of panorama. Re-parameterization can then be adjusted to match the distribution of pixels in the constructed photograph.

Upon first glance it would seem that Agarwala et al.'s [AAC+06] long scenes could be modeled by a long pushbroom projection where the path of the pushbroom corresponds to the dominant plane specified by the user. However if one examines the long scene image shown in Figure 6.8, in looking closely at the street intersections (Figure 6.9) one can see foreshortening in both the horizontal and vertical image dimensions. Such foreshortening is characteristic of perspective and is not present in a pushbroom projection where foreshortening only occurs in one dimension (perpendicular to the camera path). To create this effect with FP for such "deep areas" of the scene we adjust the parameterization of the far surface as shown in Figure 6.10 to direct projectors apart in these regions to cause desired foreshortening. In turn this causes nearby neighboring projectors to be directed more closely together; however, as long as this compression occurs only

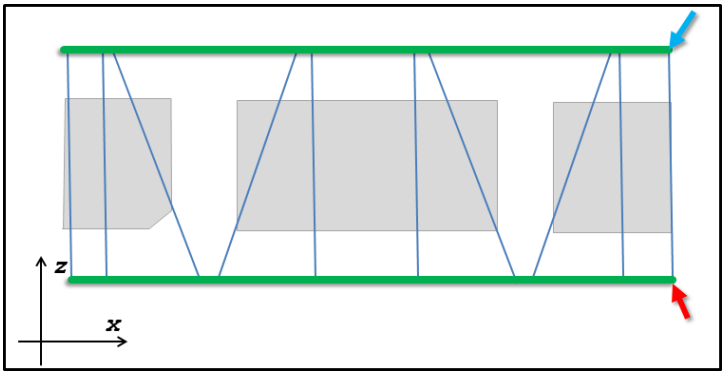


Figure 6.10: Diagram for recreating a long scene with Flexible Projection. To recreate the foreshortening down the street intersections a FP recreation can be created from two planar control surfaces. The near surface (red arrow) should feature a uniform parameterization. The far surface (blue) requires a non-uniform parameterization across its width that will compress the regions looking down the streets. The parameterization of the far surface should be adjusted to direct projectors(blue lines) as shown. In-between projectors should be interpolated.

over the shallow regions of the projection any impact that would be caused deeper within the 3D scene will remain obscured. The resulting projection created with FP is shown in Figure 6.11 and compared to a pushbroom projection.

Yu and McMillan's Multiperspective Renderings [YM04a] are created by placing compatible GLCs (i.e., GLCs whose viewing volumes fit nicely together) beside one another in a sort of image map is as shown in Figure 3.15. It has already been shown that Flexible Projection can reproduce any GLC; therefore it is clear that any Multiperspective Rendering can be reproduced by creating piecewise surfaces for the near and far surfaces. Each piece of the surface is then defined to correspond to the GLC specified by the Multiperspective Rendering's image map.

Discontinuous Multi-Camera Projections

Discontinuous multi-camera projections create collage-like blends in image-space where two objects that neighbour one another in 3D will not necessarily appear next to one another in image space. Techniques that fit this description include Digital Cubism

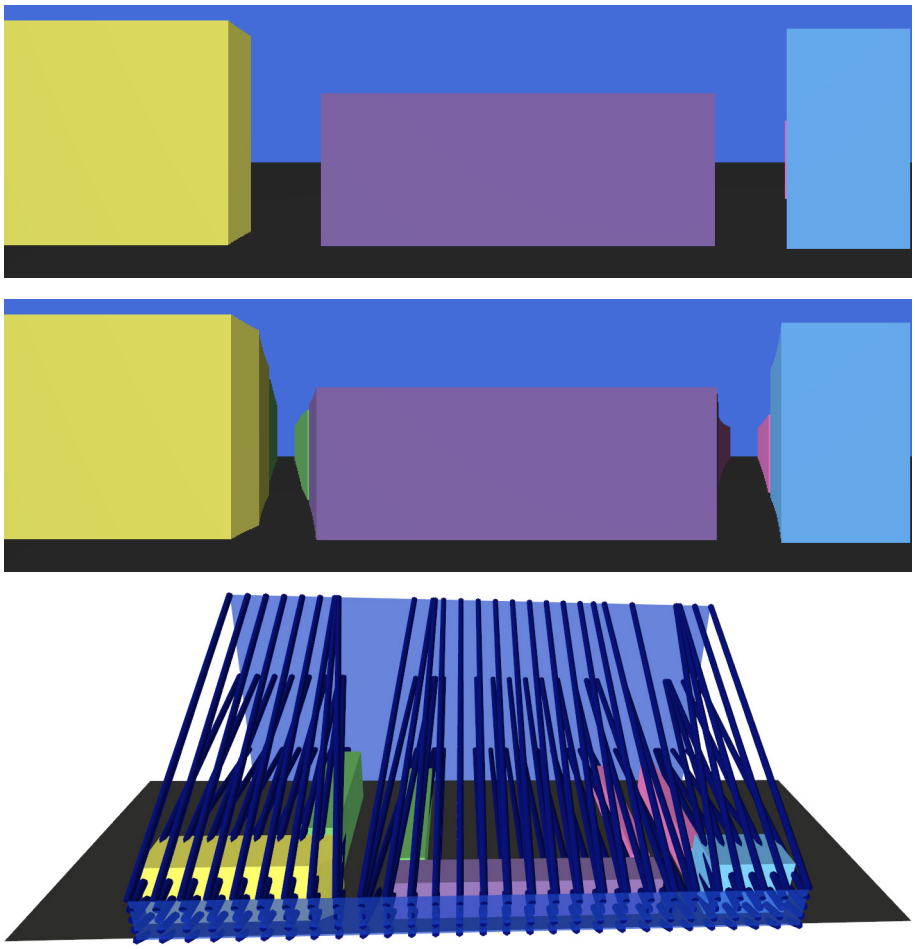


Figure 6.11: Recreating a long scene with Flexible Projection. Top: an image created with a pushbroom projection. Middle: an image created with a Flexible Projection modelled after the diagram in Figure 6.10. Bottom: scene, control surfaces, and projectors used to create the long scene like image.

[CH03], Joiners [ZMP07], the multi-camera aspects of RTCams [HCS+07], and Popescu et al.'s [PRAV09] Graph Camera. Other techniques go beyond this blending in image space and instead blend in clip space (i.e., 2.5D); such works include Artistic Multiprojection [AZM00], Fresh Perspective [Sin02], and Rendering Your Animation Nonlinearly Projected (RYAN) [CS04].

MC projections that are collaged together in image space can be recreated with a FP by creating a discontinuous volume from discontinuous control surfaces. The reparameterization operation is then used to provide a mapping from each discontinuous volume to the final image space position. An example is shown in Figure 6.12.

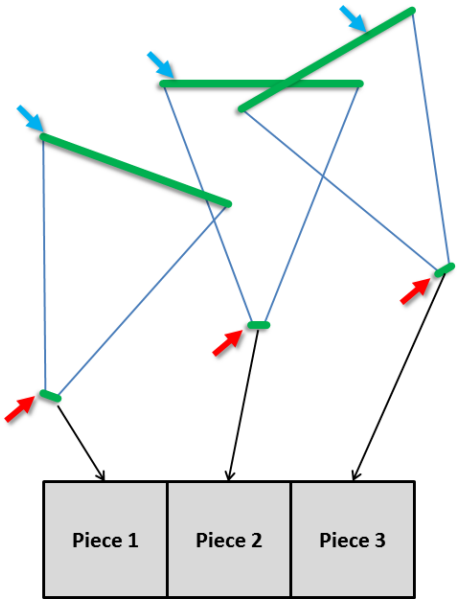


Figure 6.12: Recreating an discontinuous, collaged MC projection with FP. Discontinuous near (red arrows) and far (blue arrows) surfaces are used to create separate projection volumes, each volume contributing a piece of the resulting image. Reparameterization can be used to reposition these pieces within image space.

Handling multi-camera projections that blend in clip space is similar but more involved. For instance, consider a simple Artistic Multiprojection where we desire the majority of the scene to be projected with one perspective projection and a single object in that scene to be projected with a second perspective projection. To model this in

FP we begin with the first perspective projection's viewing volume. Then, knowing how the single object is projected in the second perspective projection we can extrude a hole through the volume from the near surface to the far, corresponding to the parts of the volume that are to be replaced by the second projection's image. The missing volume corresponding to this hole is then modeled according to the parameters of the second projection. Naturally, multiprojections that involve more than two projections would be correspondingly more complex.

Increasingly complex descriptions of how FP can be used to model these MC projections can be provided, but even from this "simple" example it is becoming clear that these descriptions are shoe-horning these MC projections into FP representations in a fashion that may not be useful. That is, modeling a discontinuous viewing volume of a MC technique is not likely to assist in addressing the main question of how the individual projections should be blended. Consequently rather than creating wildly complex description of how FP can force recreation of these MC techniques I will end this subsection with the observation that multi-camera and flexible projection provide different approaches to creating nonlinear projections, and while many, if not all, of these can be produced with FP, it might not be always useful to do so. The next section examines these two approaches more generally and describes their strengths and weaknesses relative to one another.

6.2 Comparing Discontinuous Multi-Camera and Flexible Projections

As it seems clear that FP is not able to reproduce all discontinuous MC projections in a useful fashion this section aims to provide a high level discussion and comparison of multi-camera and flexible projections.

It is first important to note that the general approach of MC projection is significantly

different than FP. The approach in MC is to find different projections that one likes to portray different parts of the scene and then attempt to blend these images together. In contrast FP attempts to model the projection as a whole, creating a single viewing volume that defines the entire projection. As a consequence, MC techniques often focus on issues such as image space blending of images and placement of each image [SS97, ZMP07] and depth determination when blending in clip space [AZM00, Sin02, CS04]. This research on FP, as demonstrated throughout the thesis, is in some ways more abstract, focusing on how to define and parameterize viewing volumes, the effects of nonlinear projectors, and the ties between the shapes of viewing volumes, projection surfaces, and the resulting image.

The remainder of this section compares the strengths and weaknesses of MC with those of FP.

6.2.1 Strengths of Multi-Camera Projection

Objects can be individually projected. The greatest strength of MC is the ability to apply and composite individual projections for numerous and complex objects. For example, in a scene with ten objects it is easier to use an MC projection to match a projection to each object than to perform this same task using an FP. Thus multi-camera projection definitions lend themselves much better to being, in a sense, object-defined.

Physical analog. A clear strength of multi-camera projections is that we can physically construct images using this technique (for example, by creating a collage or constructing a panorama from photographs). This adds a dimension of familiarity to the design process that can aid intuition. Note that this analogy does not necessarily hold for MC techniques such as those of Singh [Sin02], Coleman and Singh [CS04] where blending occurs in clip space, causing these techniques to be less easily understood.

Based on perspective projection. Due to familiarity with images created with per-

spective projection, it is often easier think of nonlinear projections in terms of small pieces of different perspective projections. That is, we can identify areas of the image that match different perspective cameras and by blending all of these identifiable pieces together we can form an idea of how the projection might be constructed as multi-camera projection.

Traditional background. Often when using multi-camera projections, objects in the foreground are individually projected and then placed in a separately projected background. This use of separate background is a common technique in traditional art it provides a good fit with the multi-camera approach.

6.2.2 Weaknesses of Multi-Camera Projection

Non-perspective cameras. Some effects are very difficult to achieve with perspective cameras. One such effect is that of inverse perspective. To allow multi-camera projections to work with other unfamiliar cameras requires some way to integrate non-perspective cameras.. Additionally, effects such as off-axis perspective or only using half of a particular camera's projected image provide increasing levels of complexity.

Diagrams are difficult. It is very hard to describe a MC projection in a comprehensible diagram (especially in 2D) thus making their creation hard to describe in books and other related media. To create a diagram for a multi-camera projection it is necessary to present the cameras including their 3D orientations, positions, and possibly their viewing frustums but also convey how the image is constructed from these camera's views. That is, one must convey how the perspective images created by the cameras are mapped into the image space as well as how these images are blended together to create a single continuous image. Lastly, variations of perspective, such as off-axis projection, can be difficult to convey in such a diagram.

Conflicts between cameras. While the concept of blending together several perspec-

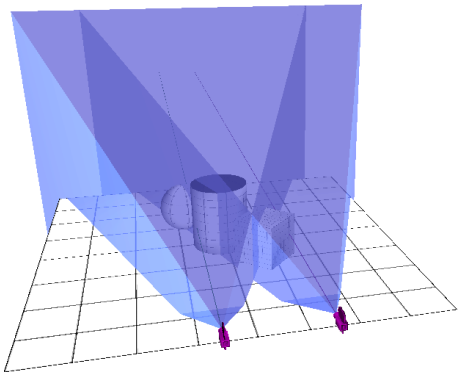


Figure 6.13: How to blend between cameras in multi-camera projections. Two perspective cameras and their viewing volumes (in blue). It is unclear what should happen in the intersection of these two volumes.

tive camera's output seems straightforward, there are problematic scenarios that occur which are hard to resolve and lack a single solution. For example, what is the desired behavior when same object is viewed by more than one camera at the same time? Consider the scenario shown in Figure 6.13 and the problem presented in its caption. An additional question is what should happen if there is an object in the gap between the two camera's viewing volumes? If we handle the transition between the volumes by interpolating camera settings as Singh [Sin02] does, should the interpolated camera show an object in the gap? Another example is two cameras at the same position but pointed in opposite directions. These ambiguous situations can be handled in a variety of different ways, however the amount of control necessary to allow users to identify and handle these scenarios in the manner they decide is difficult to provide.

Image composition requires additional control. When looking at a nonlinear projection it is possible to pick out areas of the image that conform to perspective and then estimate where cameras that could recreate the image were positioned and oriented. Consequently multi-camera projections may be perceived as easily accomplished. However, what is missing in this visualization is the process of how one goes about blending the cameras' images together.

Blending is difficult. Many problems such as overlap between cameras, or creating continuous changes between cameras' images are handled by blending images together or by interpolating between cameras. These tasks are algorithmically difficult and can be hard to explicitly control.

Lack of a viewing volume makes culling difficult to discern. Another difficulty lies in determining whether objects in the periphery of the cameras' viewing directions would appear in the composite image. Because most MCPs do not have an explicit viewing volume, culling is difficult to predict.

Global constraints. Global constraints were introduced by Coleman and Singh [CS04] to keep complicated scenes coherent. The problem is that in scenes with continuous changes between foreground, mid-ground, and background and where cameras are placed at different depths (i.e., not along a single plane), objects that are not in the direct view-direction of perspective cameras to get drawn into the composite image unexpectedly. Global constraints can be manually introduced for such objects to force them to appear where desired. Global constraints are also necessary for image-space MCP to ensure each camera's projection volume fits closely to its image space neighbour camera so that continuity is maintained in the composite image. Examples of projected images with and without these constraints are shown in Figure 6.14 as well as an FP that avoids this problem.

Many cameras are necessary to simulate a continuous volume. In order to accurately mimic projections with continuous viewing volumes, such as panoramas and fisheye projection, MCP requires a large number of cameras and, as visualized in Figure 6.15, the number of cameras required is not clear. Additionally the inaccuracies inherent in representing a continuous projection with a discrete approximation remain.

Number of parameters is non-trivial. Initially MC projections seem simple in that one only has to place cameras, rather than define a 3D viewing volume and its parame-

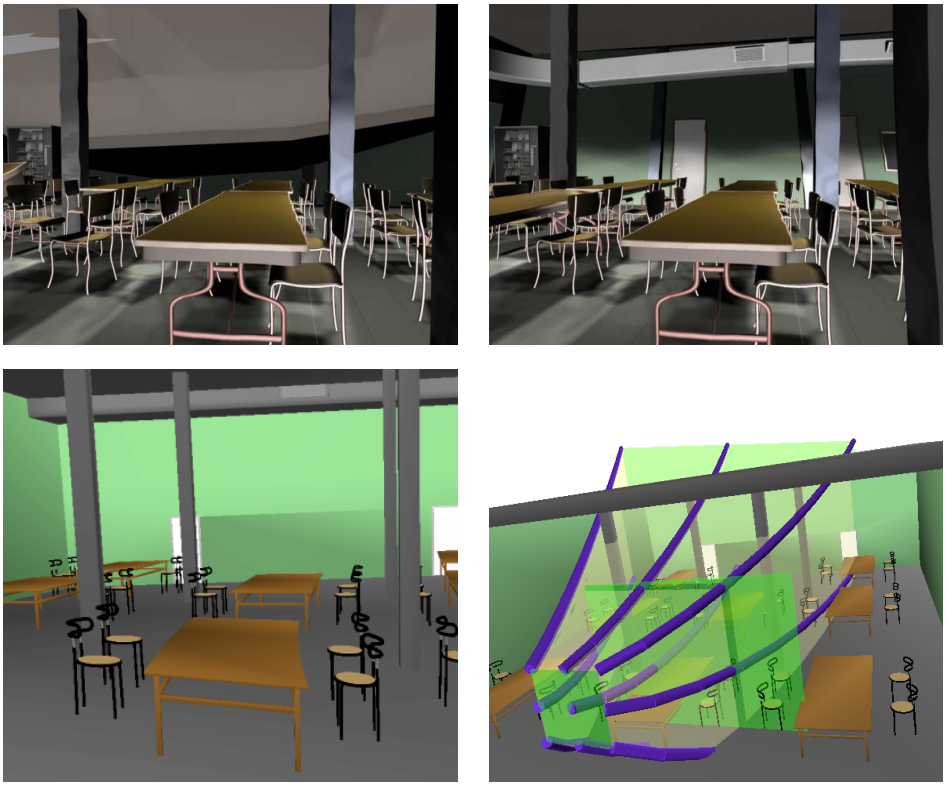


Figure 6.14: Top row: multi-camera projections from [CS04] without (left) and with (right) global constraints (images licensed for use with permission [CS04, p. 132] © ACM, Inc 2004). The global constraints prevent the ceiling from collapsing into the room. Bottom: similar image created with a FP (left) and the scene and projection settings produced the image (right). With FP global constraints are not necessary as it is possible to explicitly control the entire viewing volume.

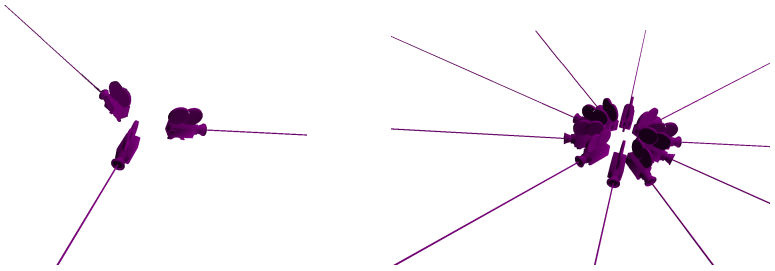


Figure 6.15: Camera positioning and orientation to create a multi-camera equivalent of a cylindrical panorama with 3 cameras (left) and 10 cameras (right). How many cameras are necessary to convey a panorama's behaviour?

terization. However, when one considers that the cameras have to be not only positioned and oriented, but also have their viewing angle set, a mapping between each camera and the image space defined, some amount of blending achieved, as well as providing global constraints the number of parameters becomes higher than one would initially expect, especially as the number of cameras grows.

6.2.3 Strengths of Flexible Projection

Fits into modeling paradigm. A major strength of flexible projection is that these viewing volumes are defined parametrically. Through this definition the volume can be controlled with existing modeling techniques in a manner that allows for deformation to desired specifications as well as interpolation over unspecified sections of the volume. These techniques, although not usually applied to projection, already exist within 3D modeling.

Continuous; built in blending. A major result of the parametric definition of viewing volumes is that the blending that is difficult to achieve with MCP, is essentially free for FP. An important aspect of this is that interpolation within a parametric volume is much more understood than interpolation between three or more cameras.

Clipping is clear. Given a defined viewing volume superimposed upon a 3D a scene it is easy to determine which parts of the scene will be present in the project image.

Curved projectors. Curved projectors allow for many possibilities in FPs. One example is in bending the projection to avoid objects that would otherwise obscure a desired element in the image.

Diagrams. It is important be able to create a diagram of a projection's settings and have the diagram accurately convey the projection's behaviour. This is extremely useful for presenting new projections in publications as well as in other environments where 3D exploration or interactive feedback is not possible.

6.2.4 Weaknesses of Flexible Projection

Volume-based behavior is unfamiliar. One of the challenging aspects of manipulating FPs is that most people are used to manipulating camera settings (i.e., position, orientation, field of view) rather than specifying a viewing volume. It takes time before an understanding of how changes to the volume affect the produced image. For example, enlarging the back surface of a volume causes far-away objects to become smaller or when curving projectors to the right causes objects to curve towards the left in the image. This is much like attempting fine motor movement while looking into a mirror; it takes some getting used to. This behavior is not unique to Flexible Projection, this same unfamiliarity is encountered when working with, for example, perspective viewing planes.

Defining 3D volumes is time consuming. The price of explicit control over the entire viewing volume is that defining some types of volumes can be time consuming. Not only does the volume have to be bound, but the projector paths through the volume also have to be defined. Some projections such as perspective and fisheye projections are easy to model, however more intricate projections such as many of the ones discussed in the previous section can be time consuming to create. This weakness may eventually be overcome through better interfaces for Flexible Projection.

Rasterizing algorithms. As will be discussed in Chapter 7 FP's realtime rendering algorithm is currently only able to handle a subset of all possible Flexible Projections. This means that often one must resort to ray tracing techniques to perform the specified projection.

6.2.5 Summary

MC and FP provide to different approaches to nonlinear projection. MC is an approach is best suited for blending several images together that excels in scenarios with large

number of objects that are projected with different projections without (or with minimal) interpolation between these projection settings. FP draws upon 3D modeling techniques, excels in producing projections with continuous changes, and can be used to describe new types of cameras.

While there is some overlap between these two approaches, they ultimately serve different purposes and should be seen as techniques that complement one another rather than as competing techniques.

6.3 Beyond Existing Nonlinear Projections

The remaining aspect regarding the Flexible Projection Framework is its ability to support exploration beyond existing projections. Consider the space of all possible single-camera projections. The related work that has been mentioned covers disjoint subsets and individual solutions within this space. We present the argument that not only does Flexible Projection cover the many individual solutions present in the existing literature, but also that Flexible Projection can interpolate between these individual solutions as well as extrapolate to new projections from existing ones.

The first way to demonstrate this is by proving that FP is capable of interpolating between existing solutions in the space of possible projections. Such a projection can be interpolated between two FPs Q_1 and Q_2 by

$$Q(u,v,t) = aQ_1(u,v,t) + (1-a)Q_2(u,v,t) \quad 0 \leq a \leq 1. \tag{6.1}$$

Such an interpolation, which is demonstrated in Figure 4.5, can be described as an interpolation between a perspective projection and a pushbroom projection.

One simple means of asserting that Flexible Projection is able to extrapolate beyond an existing projection is simply to extend the range of a in Equation 6.1. This effectively changes the viewing volume to something beyond these existing projections (Q_1 and Q_2).

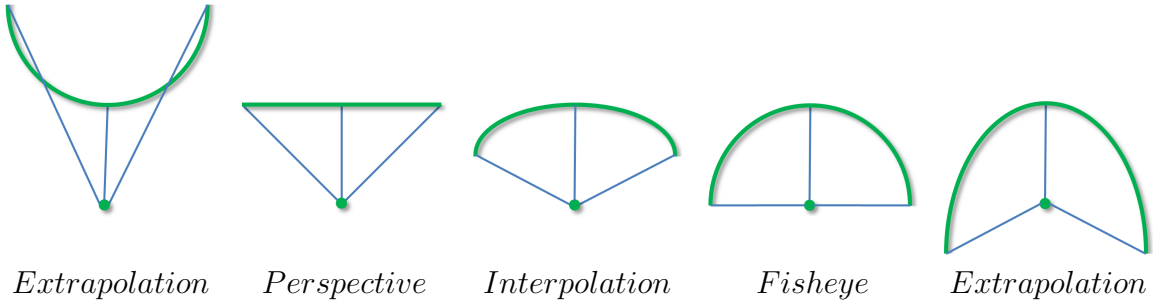


Figure 6.16: Top-down diagrams of interpolation between and extrapolation beyond a perspective and a fisheye projection.

A diagram depicting these different results is shown in Figure 6.16. While this mathematical approach to extrapolation in projection space is one possibility for developing new projections, the most-used approach in this research was to make simple changes, often transformation or deformations, to projection surfaces in known projections. For instance, the twist projection (Fig. 4.9) is the result of rotating the far surface of a perspective projection while the projection shown in Figure 4.8 is the result of bending the far surface of a perspective projection into a curve.

In general it is the ability to deform the viewing volume in a continuous manner that allows creation of an infinite set of possible projections. Additionally the creation of projections with nonlinear projectors is an area that has not been widely explored. Nonlinear projectors allow the nature of the projection to change proportionally through the depth of the projection volume. This unique capability allows for a wide variety of previously inconceivable projections to be explored.

In summary, Flexible Projection provides a thorough coverage of the assorted point solutions currently existing in nonlinear projection literature. It can be also used to interpolate between these solutions and possesses potential for supporting future exploration of nonlinear projection.

Chapter 7

Rendering

An important aspect of Flexible Projection that has not yet been discussed is how to make use of the defined projection volume to create an image. This task is an ongoing problem for all varieties of nonlinear projection as the traditional graphics pipeline does not work well with nonlinear projection as described in Chapter 2. To allow the rendering of Flexible Projections we have explored two different techniques: ray casting and an adaptation of feed forward rendering for projections that cannot be represented with projection matrices.

Ray casting Flexible Projections with linear projectors is very straightforward and can easily be extended to ray tracing. However, in projections with nonlinear projectors the projectors cannot be represented as rays; some sort of curves or piecewise linear approximation is necessary. The approach taken in this work is to limit projectors to quadratic curves. Triangle intersections with such curves can be analytically calculated with a reasonable computational burden (Section 7.2). This is a novel approach in contrast to existing nonlinear ray-tracing techniques [Gr5, Wei00] that make use of linear approximations to nonlinear projectors.

The other approach to rendering is a feed-forward approach that replaces the use of a projection matrix with nonlinear projection equations derived from the projection volume. The basis of this technique lies in extracting a projection equation that transforms camera coordinates (x, y, z) to Flexible Projection's parameterized volume coordinates (u, v, t) (Section 7.3). While such equations cannot be extracted from all possible volumes, there are several categories of volume, including some projections with nonlinear projectors, that can be rendered with this technique.

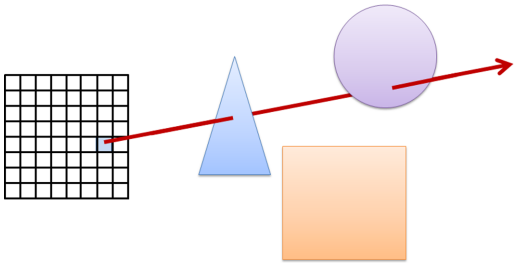


Figure 7.1: In ray casting, individual intersection testing is performed analytically between the ray (red) and the rest of the objects in the scene (the square, triangle, and circle). The material properties of the first object intersected are used in lighting calculations to determine the colour of the pixel in the image (grid on left).

7.1 Ray Casting

Ray casting techniques create images by simulating the flow of light rays. The basic algorithm is, for each pixel in the image, create a ray with its origin at the corresponding position on the near surface and directed toward the corresponding position on the far surface. The next step in the algorithm is to test each object in the scene to see if it intersects the ray as shown in Figure 7.1. Once the intersection closest to the near surface is found, lighting calculations are performed based on the material of the object intersected. Ray tracing [Gla89] refers to the more complicated version of this algorithm where secondary rays are generated, usually at the point of intersection, and intersected with the scene to allow simulation of effects such as reflection, refraction, and shadows. Image creation with ray casting or ray tracing generally results in high quality images compared to feed-forward rendering approaches but, for complex scenes, these techniques are not usually capable of image generation at speeds necessary for interactive applications.

Flexible Projections with linear projectors (e.g., the projections described in Section 4.1) can be rendered with any standard ray tracer. The projection framework is merely used to position and orient the rays that sample the scene. To position and orient a ray, we specify the image space coordinate (x, y) corresponding to the desired ray. Next, re-parameterization is used to map these pixel coordinates to the surface coordinates

(u, v). The ray that originates at (u, v) is defined by the projector:

$$q_{u,v}(t) = (1 - t)Q_n(u, v) + tQ_f(u, v).$$

Intersection tests occur in the same manner as in conventional ray tracers where $t \in [0, 1]$ denotes valid intersections occurring within the viewing volume. Other details such as lighting, anti-aliasing, acceleration techniques, reflection, refraction, and other operations remain the same as in conventional ray tracers including the significant amount of time necessary to rendering large and complex scenes.

7.2 Nonlinear Projector Casting

Analytically it is clear that the problem of nonlinear projector-object intersection can be much harder than linear projector-object intersection as illustrated in Figure 7.2. Our overall approach is to limit our system to quadratic curves where it is still relatively inexpensive to analytically calculate these intersections. Although this results in projections that are less complex than those with higher order projectors, it appears that quadratics are powerful enough to create a wide variety of interesting projections. The details for creating an intersection test between quadratic curves and triangles is described in Subsection 7.2.1.

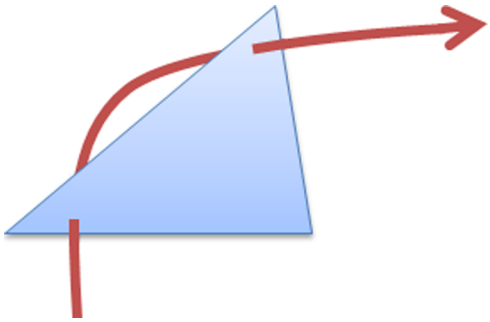


Figure 7.2: When ray casting with curved rays care must be taken regarding multiple intersections. That is, even for simple objects such a triangles curved rays may intersect the same triangle more than once.

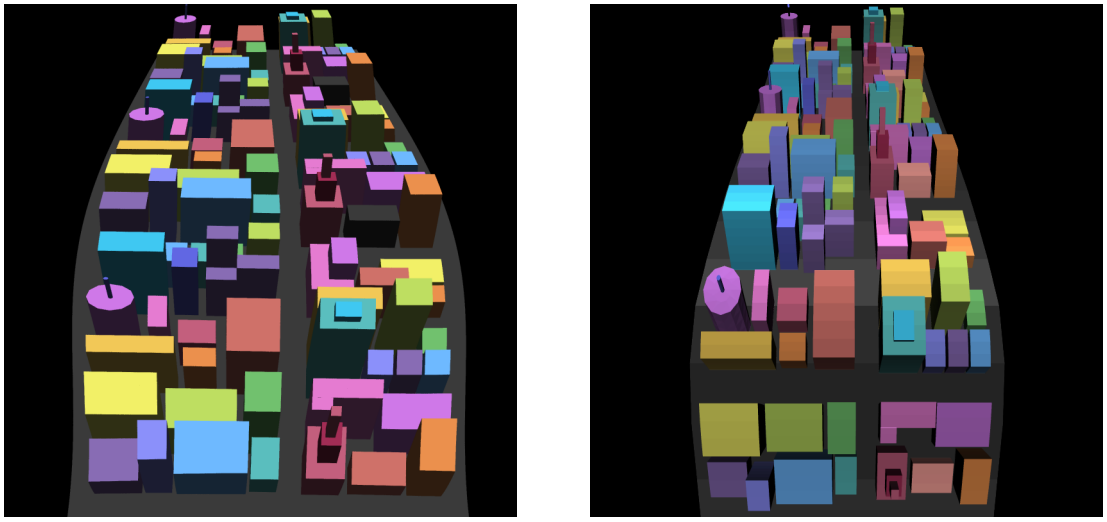


Figure 7.3: In the left image the lighting is based on the model's original geometry, this constancy is noticeable in that all the buildings appear lit from above their roofs. In the right image, lighting calculations have been performed after the transformation into two and half dimensions (i.e., image space with depth) making the lighting direction appear to change through the image. This change in lighting seems to indicate the object was deformed rather than nonlinearly projected.

The alternative implementation is to use many small rays to approximate each nonlinear projector. This approach is taken by existing nonlinear ray tracing techniques [Gr5, Wei00, WSE04] where the projections implicitly defined by underlying dynamical systems lacks any possibility of assuring the generation of low order curves. This is opposed to Flexible Projection where projector curves are explicitly defined. Approximation by linear segments is useful and most accurate when the nonlinear projectors do not deviate greatly from linear projectors. Highly curved projectors require a large number of ray segments to approximate the curve in a useful fashion and consequently feature higher computational costs to maintain accuracy. An initial examination of the trade-offs between tracing quadratic curves against piecewise approximations is described in Subsection 7.2.2.

A difference between nonlinear projector casting and ray casting is in performing lighting, reflection, or refraction calculations when using curved projectors. A relevant

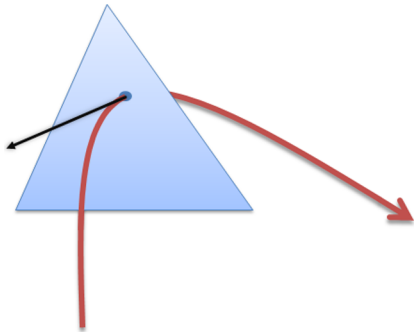


Figure 7.4: When performing lighting calculations with the ray casting algorithm the tangent of the nonlinear projector (red) at the point of intersection (dark blue) acts as the viewing direction (black).

reminder is that projection refers to control over the transformation from three dimensions to two dimensions. Thus projection should not affect the existing lights or shadows present in the scene (see Figure 7.3); to do so would be deformation rather than projection. Consequently in these calculations we use the original object normals present in the scene's geometry. Calculation of specular highlights and reflections require a direction to the viewer. As no single viewing direction may exist we estimate the projector's tangent at the point of intersection as shown in Figure 7.4. This is a reasonable approach as it takes into account the path of the projector at the point in space where it intersects the object.

7.2.1 Raycasting Quadratic Curves

The basic operation required to ray cast quadratic curves is the quadratic-triangle intersection test. The key idea behind this test is the use of the quadratic root-finding equation to calculate the point on the curve that intersects the plane of the triangle. This subsection first describe the calculations necessary to determine whether a quadratic curve intersects a plane, and then describes a quadratic-triangle intersection algorithm.

Quadratic-Plane Intersection

The quadratic root-finding equation

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

provides strong analytic solution for determining which values of t cause a quadratic curve $at^2 + bt + c$ to evaluate to zero ($a, b, c \in \Re$). The general strategy for this intersection test is to alter our curve and plane to change the problem to that of root-finding so that this equation can be employed.

We accomplish this change by rotating the plane so that it is perpendicular to the x-axis and then translating the plane by its x intercept so that it includes the origin. Then we apply this rotation and translation to the curved projector. The roots of the transformed curve mark the intersection points between the curve and the YOZ plane and, when these points are transformed by the inverse of the rotation and translation, the roots mark the intersections between the original curve and the plane.

Once the rotation R and intercept x_{int} have been calculated we can find the roots of our quadratic with:

$$RQ_{u,v}(t) - x_{int},$$

where $Q_{u,v}(t)$ is our quadratic projector, R is the rotation matrix, and x_{int} is the x-intercept of the rotated plane. The projector's equation then must be altered to the $at^2 + bt + c = 0$ form used by the quadratic root-finding equation. This calculation will depend on the formulation of $Q_{u,v}$. To provide an example, if we assume $Q_{u,v}$ is the Bézier curve

$$(1-t)^2 P_0 + 2(1-t)P_1 + t^2 P_2$$

where P_i is the i^{th} control point of the curve. Then the rotated version of the curve is

$$RP_{0_x}(1-t)^2 + RP_{1_x}t(1-t) + RP_{2_x}t^2 = x_{int},$$

which can then be put in the form $at^2 + bt + c = 0$

$$[R(P_{0_x} - 2P_{1_x} + P_{2_x})]t^2 + [2R(P_{1_x} - P_{0_x})]t + (RP_{0_x} - x_{int}) = 0$$

with which we can use the quadratic root-finding equation to solve for t.

This technique appears expensive considering that this test is performed m times for each quadratic being traced (where m is the number of planes in the scene). It can be made less costly by noting that the plane's rotation and x-intercept can be pre-calculated and stored, rather than be recalculated for each intersection test.

When implementing this test, is important to ensure that special cases such as non-real roots (i.e., $b^2 - 4ac < 0$) and linear curves (i.e., $a = 0$) and handled carefully.

Quadratic-Triangle Intersection Test

The first step in performing the quadratic-triangle intersection test is to find intersections between the quadratic curve and the plane defined by the triangle. Once this has been done we are left with zero, one or two points where the quadratic curve intersects the plane of the triangle.

These point(s) are then tested to see if they lie upon the segment of the curve that is within the volume volume. Next the point(s) of intersection's barycentric coordinates are calculated relative to the triangle (see [Len04] for an efficient example) to determine whether the point(s) lies within that triangle. In the scenario where both points pass this test (i.e., both roots represent points that are both within the volume and lie on the triangle) the algorithm returns the point that is closest to $Q_n(t)$, along the path of the curve.

Pseudo code for this test and the quadratic-plane intersection test is listed in Appendix B.

Model	No. Faces	Quadratic	2 Segments	3 Segments	4 Segments	5 Segments	6 Segments	Ratio
Kia	520	72.74	48.56	59.73	69.98	84.99	114.57	0.5
Teapot	2256	326.06	185.99	296.43	380.10	433.48	474.40	0.66
Cowboy	7695	1167.76	875.53	1261.39	1640.88	1807.96	2078.57	0.3
Alley	12602	1711.28	1342.75	1604.97	2192.11	2583.69	2891.87	0.8
City	22123	3494.66	1837.91	2537.59	2862.20	3345.76	4554.37	0.5
Saloon	37127	4642.61	3235.35	5075.96	6175.63	6810.99	7239.51	0.5

Table 7.1: Timing data comparing quadratic casting to piecewise raycasting of 2-6 segment linear rays of an 800 by 800 pixel image. N segments refers to casting of rays with N linear segments. All timings are in seconds. *Ratio* refers to the approximate amount of the image that was occupied by the rendered object.

7.2.2 Performance

To ascertain the usefulness of quadratic casting, in comparison to piecewise ray casting, an initial timing test has been performed.

This test compares the time to ray cast an image, using the quadratic casting algorithm described in the previous section, to an algorithm that makes use of piecewise linear ray casting. The piecewise linear algorithm used in this testing performs a standard ray-triangle intersection test based on dot-product projection [Len04]. The piecewise algorithm performs this intersection test on each piecewise segment of the ray, beginning with the segment touching the near surface. If no intersection is found with the first segment, the algorithm tests for intersection with the next segment, continuing until either an intersection is found or the last segment has been checked. If an intersection is found, intersection tests on subsequent segments are not performed. Consequently, the intersection tests will be completed more quickly for piecewise rays where an intersection occurs close to the near surface than situations where intersections occur later or no intersection is found. No acceleration techniques (i.e., bounding boxes, space trees, GPU based implementation) were used for either algorithm.

For each test a single model was placed within the middle of the viewing volume and an 800 by 800 pixel image was traced. The viewing volumes used for each type of a projection were similarly constructed, essentially forming a perspective volume with an

upward bend through the volume. This shape was chosen to ensure that the quadratic projectors did not revert to linear curves at any time. The linear ray segments formed piecewise linear curves with each segment of a similar length.

The results are shown in Table 7.1. In general, these timings feature high variability. Both algorithms can be unpredictably affected by the specified viewing volume, the exact nature of the intersection (e.g., whether the quadratic curve intersects the triangle's plane zero, one, or two times), the number of rays that do not intersect any object (i.e., meaning that all segments of the piecewise ray will have to be tested), etc. Additionally, errors were likely introduced by background PC processes. However, despite these qualifications, the results do provide an initial estimate of the relative performance of the quadratic raycasting algorithm.

The results shown in Figure 7.5 illustrate that quadratic ray casting is generally more expensive than casting piecewise rays of two segments but less expensive than piecewise rays of six segments; three, four, and five segment piecewise rays may or may not take longer than quadratic casting, depending on the previously mentioned conditions. Figure 7.6 illustrates the change in rendering time relative to increases in model size, indicating that all tests portray a roughly linear increase in time with an increase in the number of triangles.

This initial test indicates that quadratic ray casting holds great potential for efficient ray-casting of curves. In scenarios where more than five piecewise segments are required to accurately reproduce a quadratic curve, performing direct intersection with the curve is likely to be more efficient. One can imagine a hybrid algorithm that makes use of Gröller's hierarchical approximation of piecewise rays [Gr5] to determine how many piecewise segments are required to approximate a quadratic ray. This can then be used to choose the faster, nonlinear ray casting algorithm.

There are factors that can change this calculation. For instance there exists a wide

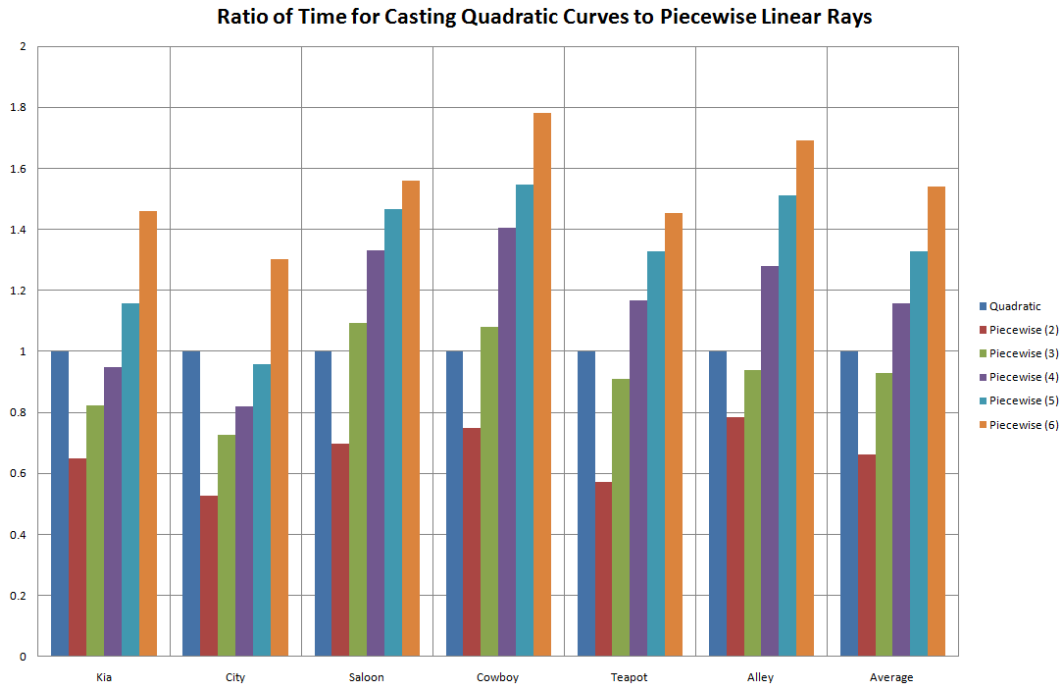


Figure 7.5: Bar graph of the ratios of the time required to render each image to the time required for the image created by casting quadratic rays. The x-axis labels identify the model used.

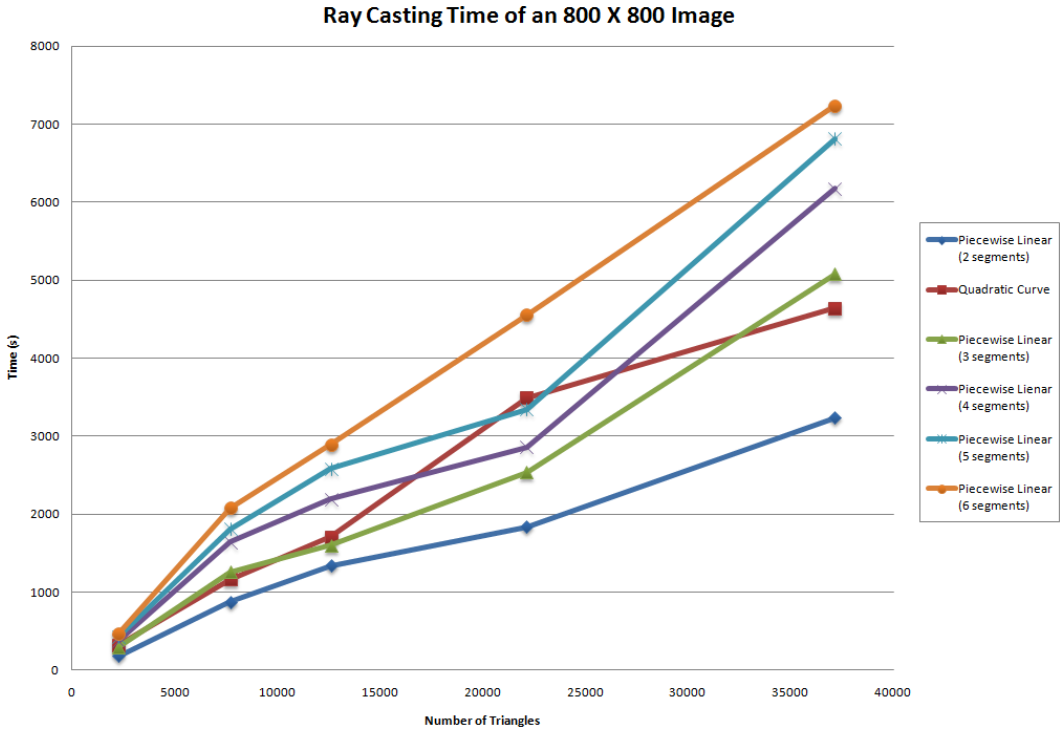


Figure 7.6: Line graph comparing the ray casting time to the number of triangles in the model.

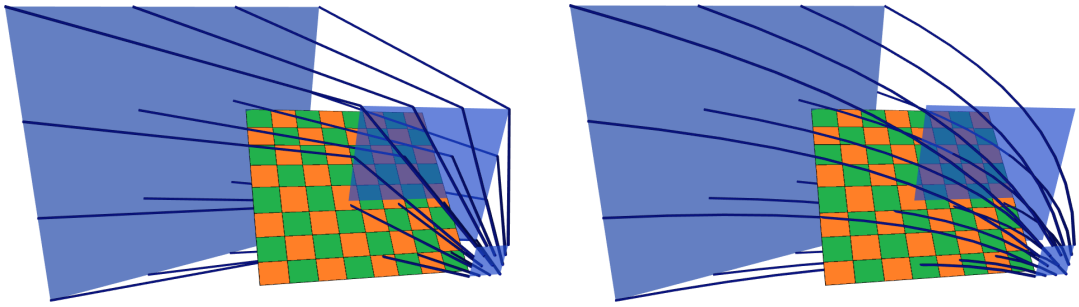


Figure 7.7: A transition in a projection using piecewise linear (left) and quadratic (right) projectors.

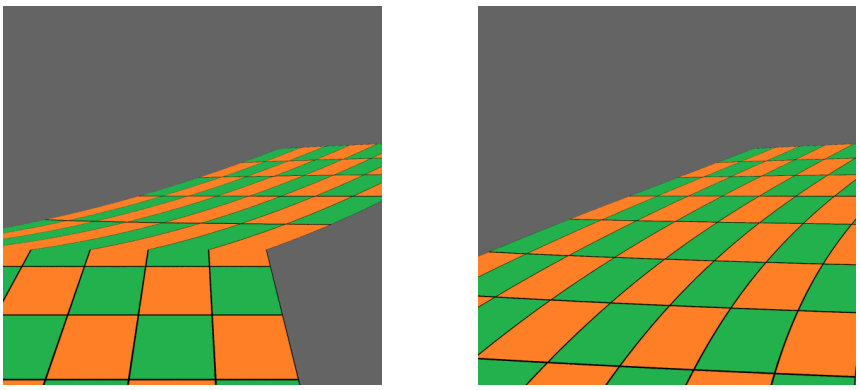


Figure 7.8: Images created from the volumes shown in Figure 7.7.

variety of acceleration techniques such as spatial subdivision and GPU-based ray casting that may or may not be adapted to tracing quadratic rays. Such techniques greatly increase the speed of ray casting but research is required to see if these techniques can be effectively adapted for quadratic rays. Another consideration is extension of the quadratic tracing algorithm to higher order curves. That is, piecing together quadratic curves may prove a more efficient approach to ray casting higher order curves when compared to direct analytical tests or compared to the use of many linear curve segments.

Another consideration for the use of quadratic projectors is their ability to blend between different depth-wise regions of the projection. Consider the scenario where we wish to have one perspective projection at the shallow depths of the volume and another at the end. We can perform this transition in one of two ways; either with piecewise

linear rays (Fig. 7.7 left) or with a quadratic projector (Fig. 7.7 right). Looking at the resulting projections (Fig. 7.8), we can see that the piecewise linear projectors cause a sharp transition in the projected image. The quadratic projectors create a smooth change between the projections. Consequently, it is clear that a quadratic projectors provide a useful impact in image creation that would require a large number of linear projectors to approximate.

This section has outlined a technique for rendering polygonal models through an analytic triangle-quadratic curve intersection test. Although these casting and tracing techniques are very accurate and able to reproduce a wide variety of photorealistic phenomena, they remain a time consuming process that only achieves interactive frame-rates with simple scenes or with extremely powerful hardware. The next section explores a more limited feed-forward scanline rendering technique that achieves interactive and real-time frame-rates on relatively standard graphics hardware.

7.3 Scanline Rendering Algorithm

Scanline rendering techniques are generally limited to triangles. To render triangles they use the projection matrix to transform each triangle's vertices from camera coordinates to normalized device coordinates. After the clipping algorithm removes unnecessary triangles and pieces thereof, the scanline algorithm draws the triangles into the frame buffer line-by-line based on bi-linear interpolation and the transformed vertices.

In order to use this technique to render Flexible Projections we must replace the use of the projection matrix with a nonlinear projection equation derived from the viewing volume. This derivation is non-trivial and is the primary focus of this section. The replacement of the projection matrix causes all triangle vertices to be moved to positions dictated by the nonlinear projection. However, the pixels that make up the triangle

Figure 7.9: Comparison between feed-forward rendering and ray-casting of a fisheye projection. The left and center images were produced with the feed-forward rendering technique. The left image displays the artifacts present when a low-polygon count model is used; the center image was created with a high polygon count version of the model. The right image was created with ray-casting.

are still rendered by scanline drawing algorithms that assume the triangle has been linearly projected and thus has straight edges and that its properties can be accurately interpolated with linear interpolation. Since this is not the case after a triangle has been nonlinear projected, our adaptation of these algorithms does result in inaccuracies. This problem can be alleviated by subdividing the models in the scene. An example of these artifacts in a coarse and refined version of a polygonal mesh is shown in Figure 7.9. Such inaccuracies are also encountered by other nonlinear projection feed-forward based rendering techniques such as the multi-camera technique of Coleman and Singh [CS04].

To re-phrase this problem, the primary task of our scanline algorithm is, to find a projection equation that, for each vertex of the scene, will change its spatial representation (x, y, z) to a parametric representation (u, v, t). Notice that this is the inverse of our parametric volume $Q(u, v, t)$. Re-parameterization provides an additional step where the u and v values are transformed into pixel coordinates.

As mentioned the derivation of a projection equation from a Flexible Projection viewing volume is a challenging problem. These equations are not easily calculated or even possible for all viewing volumes. For instance if $Q(u, v, t)$ is not one-to-one then for some points Q^{-1} will have to yield more than one value, a problematic scenario.

So, keeping in mind that we will not be able to find a projection equations for all possible volumes, the first step in our approach is to limit ourselves to volumes where t can be calculated efficiently given $Q(u, v, t)$ and the point we wish to project $p = (x, y, z)$. Simple examples of where this is possible are the linear projections shown in Figure 4.3. For these projections, t can be calculated by the ratio of the distance of p to Q_n to the distance between Q_n and Q_f. We can extend this to all volumes where, for any two values of t, t_1 and t_2, $Q_{t_1}(u, v) - Q_{t_2}(u, v)$ is constant for all values of u and v. Examples of volumes that meet this criteria include volume composed of planar control surfaces that share the same normal (e.g., Fig. 4.9), or volumes created from nested spheres or cylinders (e.g., 4.7). These projection may have linear or nonlinear projectors and may be linear or nonlinear projections. Other projections that meet this criteria shown thus far in the thesis include Figures 4.5, and 4.12.

Note that t must be calculable in an efficient manner, since for realtime rendering t will have to be calculated for every vertex in every frame. Figure 4.8 provides an example of a projection where t is not easily calculated and thus, to our knowledge, cannot be implemented with this scanline technique with realtime results.

Now that we have found t', the particular depth value of p within the projection volume, the problem has been reduced to solving for u and v. With the description of the volume $Q(u, v, t)$, we are able to find the iso-surface $Q_{t'}(u, v) = Q(u, v, t')$ that p is located on. The next step is to determine the u, v coordinates for p on $Q_{t'}(u, v)$. The exact details of this calculation depend on the definition of $Q_{t'}(u, v)$. A diagram outlining this process is shown in Figure 7.10.

Our description to this point has been applied to Flexible Projections in general; however within the specific implementation, for instance when using control surfaces, there are some additional considerations. Firstly, if the control surfaces are all the same type of parametric surface, then $Q_{t'}(u, v)$ is easily extracted from $Q(u, v, t)$ as a surface

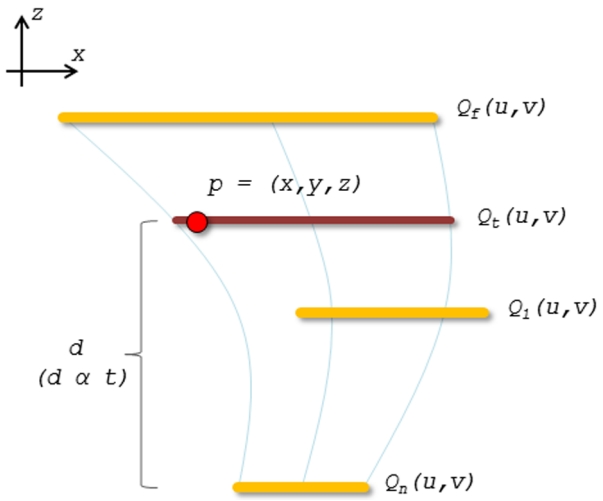


Figure 7.10: Finding (u, v, t) given $p = (x, y, z)$ within a viewing volume defined by three surfaces $Q_n(u, v)$, $Q_1(u, v)$, and $Q_f(u, v)$ (orange lines) that create quadratic Bézier curves as projectors (thin blue lines). The technique relies on d being proportional to t. Once t' is calculated, the iso-surface $Q_{t'}(u, v) = Q(u, v, t')$ (red line) that p lies upon can be extrated. Then a surface-specific calculation that depends on the parameterization of the surface is performed to find $u,\ v$ such that $Q_{t'}(u, v) = p$.

of this same type. Next, we should also choose surfaces where, given a point p on the surface $Q(u, v)$ one can calculated $u,\ v$ such that $Q(u, v) = p$. As a simple example, consider the parametric cylinder

$$Q(u,v) = \begin{bmatrix} R\cos(u) \\ \frac{1}{2}Hv \\ R\sin(u) \end{bmatrix} \quad 0 \le u \le 2\pi, -1 \le v \le 1$$

where H and R are the height and radius of the cylinder. With this surface we can easily calculate u by determining the angle of the vector (x, z) and v by comparing y with H.

Altogether this process for determining t and then u and v provides our projection equation $Q(x, y, z) = (u, v, t)$. The remaining task is to ensure that, if this equation is nonlinear, that it is applied to each vertex being rendered, instead of a projection matrix. While this can be done the CPU (central processing unit), on current graphics hardware this replacement is much more easily and efficiently implemented using programmable

shaders, particularly the vertex shader. On the vertex shader it is straightforward to, after model, world, and camera transformations have been applied, override the projection matrix transformation with a nonlinear projection equation. This allows the nonlinear projection to be performed on vertices in parallel across hundreds of cores on the GPU (graphics processing unit) rather than the relatively few cores of the CPU.

In summary, scanline rendering requires a projection equation $Q^{-1}(x, y, z)$ that is non-trivial to derive for an arbitrary projection volume $Q(u, v, t)$. Our process to derive this equation is to limit the projection volume to those that:

1. allow t to be easily calculated given $Q(u, v, t)$ and p,
2. are defined by parametric surfaces of the same type,
3. and this type of parametric surface $Q(u, v)$ possesses a means of solving for $u,\ v$ given p on the surface.

In Section 9.4 we provide a detailed description of an extension to this technique that allows a simple numerical approximation to assist in solving for $u,\ v$ (the third step listed above).

Chapter 8

Applications

To demonstrate Flexible Projection's overall usefullness and adaptability, this Chapter presents several different applications that would be difficult, if not impossible, to achieve without this framework.

Section 8.1 describes animated cameras. These are cameras where the type of projection smoothly change over the course of an animation. While interpolation of projection parameters is possible with projection matrices, the modeling of the projection volume used in Flexible Projection greatly expands the possibilities of these transitions, allowing interpolation between extremely different types of projection.

Another application of Flexible Projection is described in Section 8.2 where it was used with an art installation. Flexible Projection provided a means to create a variety of novel views and outlooks upon a virtual environment.

Lastly, in Chapter 1 the possibility of recreating the nonlinear projections used by artists was mentioned as motivation for Flexible Projection. Artists frequently break the rules of perspective, subtly altering the image. Such projections are very difficult to accomplish with projection matrices but this reproduction becomes possible with Flexible Projection as is described in Section 8.3.

8.1 Animated Cameras

Control over the camera is crucial to cinema as well as virtual experiences created through CG animation and video games. Changes in zoom, panning, dollying, and trucking are well known effects in cinema and animation [WFH+97]. Novel effects, such as the "bullet

time" effect [Wik10] created in *The Matrix* [WW99] that allows the audience's point of view to move while events in the film are slowed, can provide movies and other media with a wow factor and allow previously impossible visualizations to be used for great effect. In this section we introduce the concept of an animated camera, a camera that can gradually change the structure of its projection over time. This change in projection allows for a variety of new effects and viewing changes to be considered.

The advantage of performing such transitions with Flexible Projections as opposed to projections defined in another fashion is that Flexible Projection volumes have a regular structure that directly maps onto normalized device coordinates. This makes interpolation between two different projection's viewing volumes very easy, especially when compared to the general problem of morphing between arbitrary volumes. The possibilities inherent with and the ease of these transitions will be shown through three examples: a new type of camera rotation (Sub-section 8.1.1), bending the projection volume to reveal obscured objects (Sub-section 8.1.2), and lastly altering the projection to focus the viewer's attention (Sub-section 8.1.3).

8.1.1 Rotation

One possibility for animating projections exists in changing ordinary camera movements into changes in the structure of the applied projection. The particular movement we will demonstrate in this scenario is rotation around the camera's up axis. As an alternative to standard rotation, one can create a staggered rotation where the camera, when rotated, causes objects near the camera to come into view first while objects that are further away appear later. One can imagine that such an effect could be used subtly to provide cues that the depicted environment is unreal (such as in the movie *Inception* [Nol10]), to indicate the narrator is inebriated, or to emphasize other movement within the scene.

To create this staggered rotation projection we begin with a perspective projection

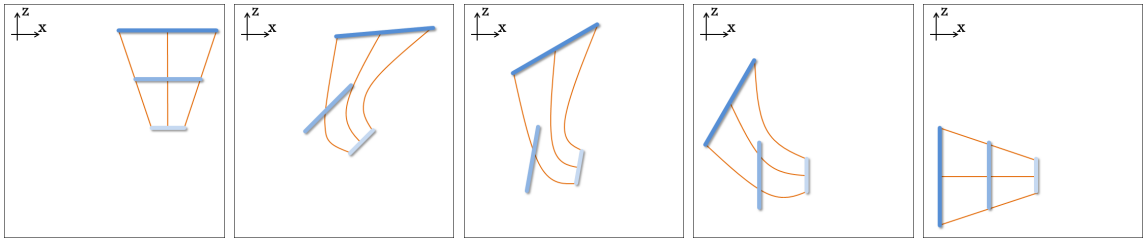


Figure 8.1: Movement of three control surfaces (blue) in creating a staggered rotation.

created from three surfaces; the usual near and far planes as well as a middle surface created by interpolation between the near and the far surfaces. In general to create the staggered rotation effect we first rotate the near and middle surface together, while leaving the far surface in its original position. This causes the parts of the scene close to the near surface to rotate into view while the background of the scene remains static. As shown in Figure 8.1, only when the rotation of the near and middle surfaces is almost complete, will the far surface begin to rotate and eventually align with the other surfaces.

The middle surface must be positioned carefully between the near and the far surface as shown in Figure 8.2. If the middle surface is too close to the far surface the projectors' bend toward the far surface will occur outside the scene and the far surface's impact will not be noticeable. If the middle surface is too close to the near surface the projectors will bend towards the far surface before any of the scene's geometry can be intersected and the near surface's impact will not be noticeable. The ideal scenario is to position the middle surface so that bend in the projectors occurs through the scene's geometry.

A diagram comparing 90°s of rotation of a usual, uniform rotation to a staggered rotation are shown in Figures 8.3, 8.4, and 8.5. The amount of exaggeration present in this effect can be controlled by adjusting when the far surface begins moving. Initiating the rotation of the far surface earlier will make the resulting effect less noticeable.

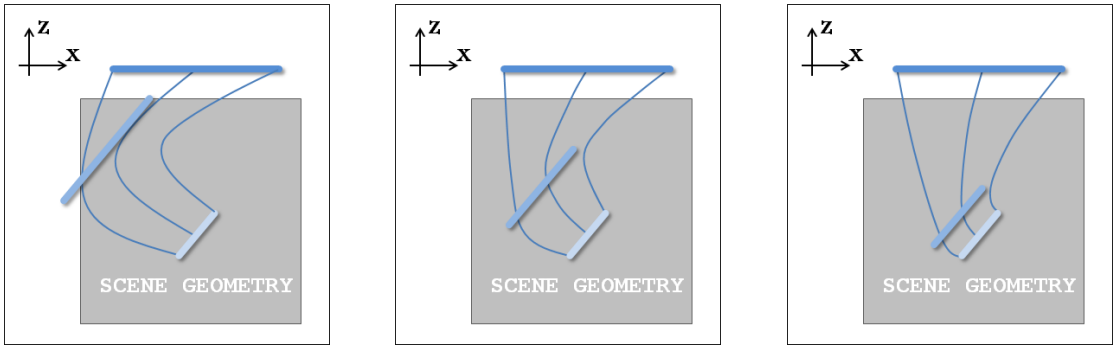


Figure 8.2: Positioning the middle surface. Left: middle surface is too far from near surface. Center: middle surface has been positioned to make projectors bend through the scene's geometry. Right: middle surface is too close to the near surface.

8.1.2 Avoiding Occlusion

One apparent use for curved projectors in Flexible Projection lies in bending the viewing volume around obstacles to expose obscured objects. However, if we just present this unobscured view the viewer is likely to be confused about the composition of the scene because they cannot be sure of the relation of what they are seeing to the perspectively projected scene. Use of an animated camera provides a sort of temporal coherence for this task, making the changes in the viewing volume and exposure of hidden objects much more understandable.

In an example shown in Figure 8.6 the viewer is stuck in traffic and unable to determine the cause of a traffic stoppage. The projection has then been altered to remain from the same viewpoint but curved around the obscuring cars to reveal the problem.

The initial projection used in this example is a three plane perspective projection. The projection that avoids the occlusion is created by moving the middle plane to the left and downward while the far plane is moved slightly to the right (as shown in Figure 8.7). In-between frames created by interpolating between the perspective and the altered projection's viewing volumes are shown in Figure 8.8.

Another possibility with occlusion is the scenario where we wish to see through a

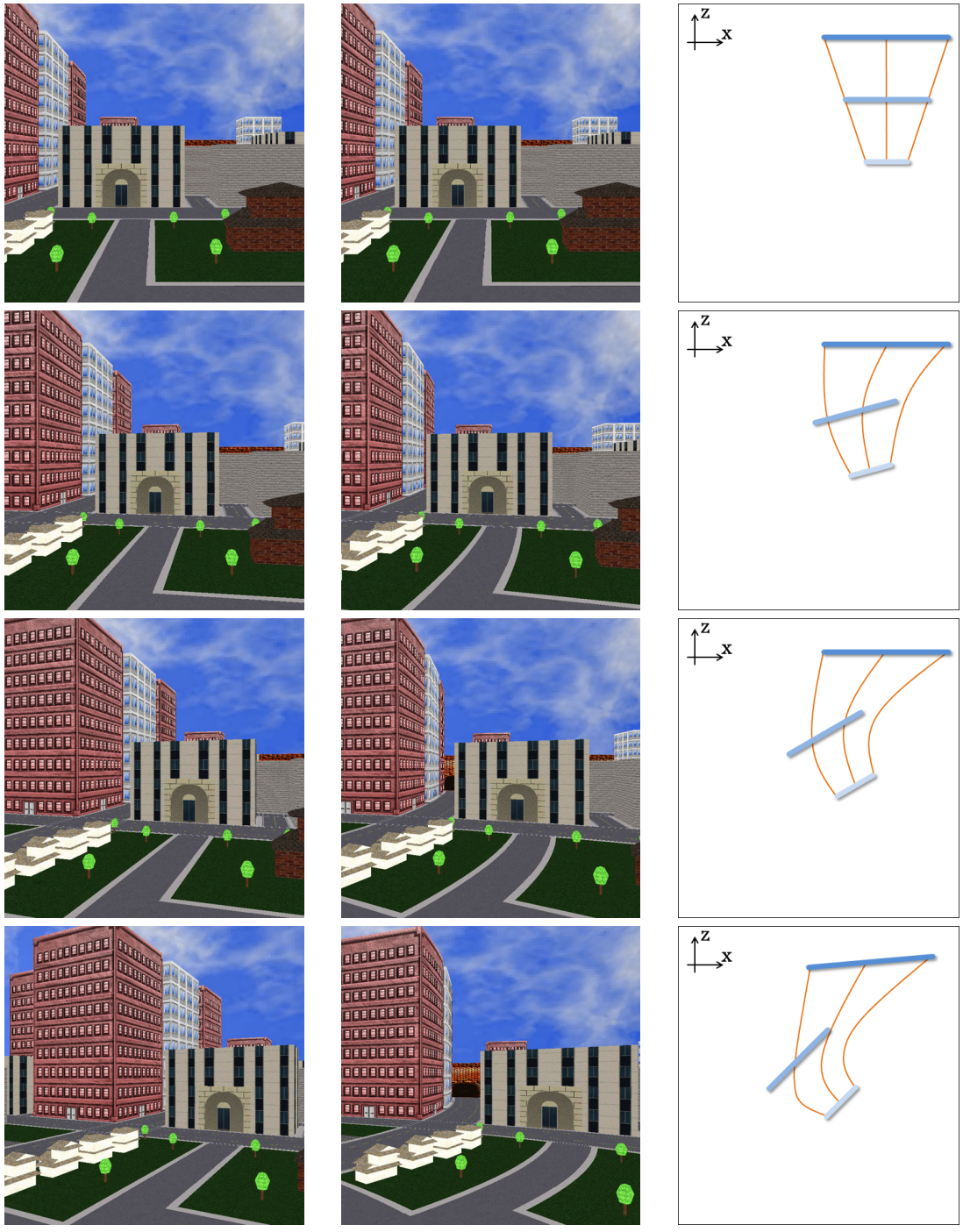


Figure 8.3: Comparison between a camera that uniformly rotates 9° per frame (left) and a staggered camera (center). The images on the right show the top-down positions of the surfaces used to created the staggered camera (1/3).

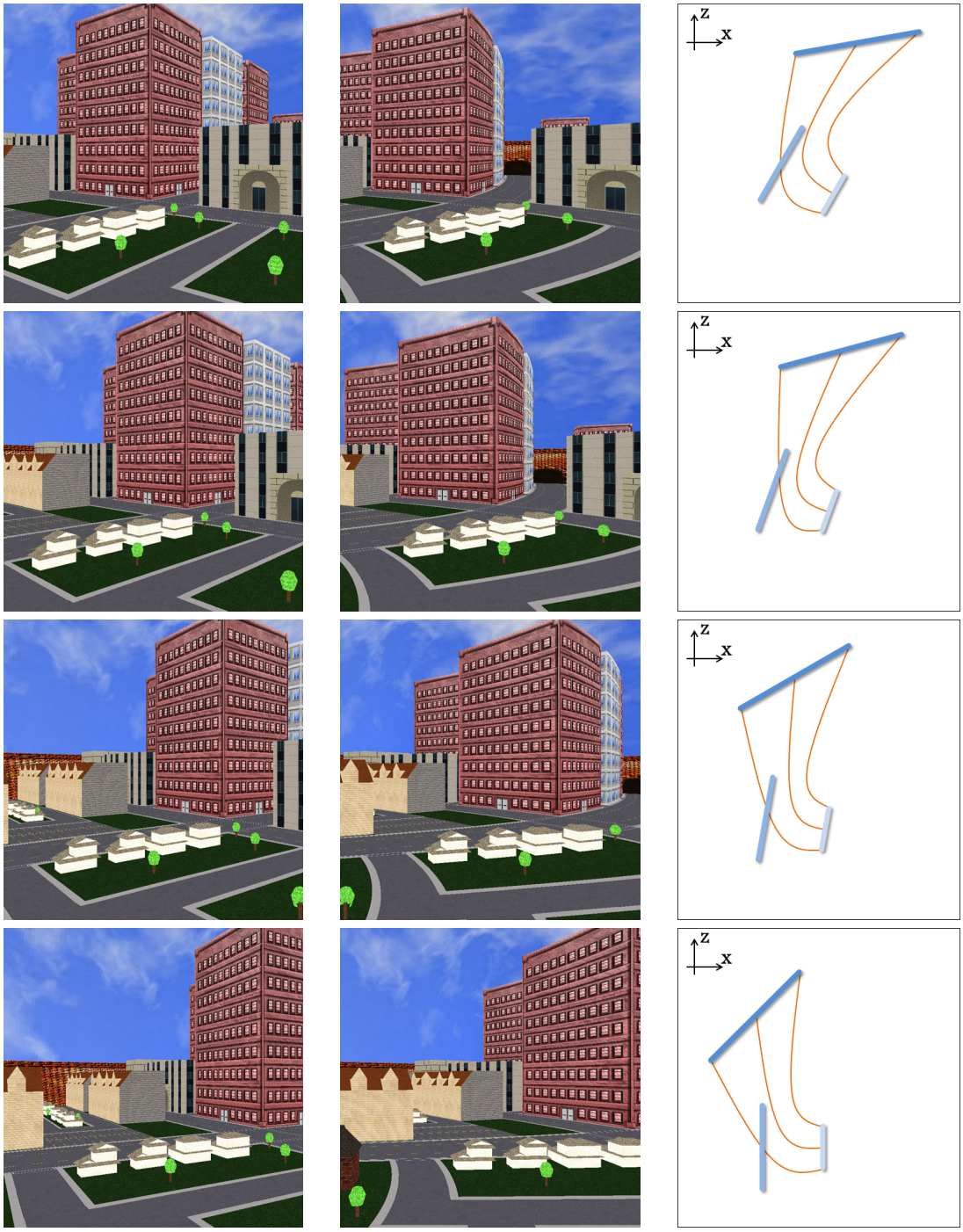


Figure 8.4: Comparison between a camera that uniformly rotates 9° per frame (left) and a staggered camera (center). The images on the right show the top-down positions of the surfaces used to created the staggered camera (2/3).

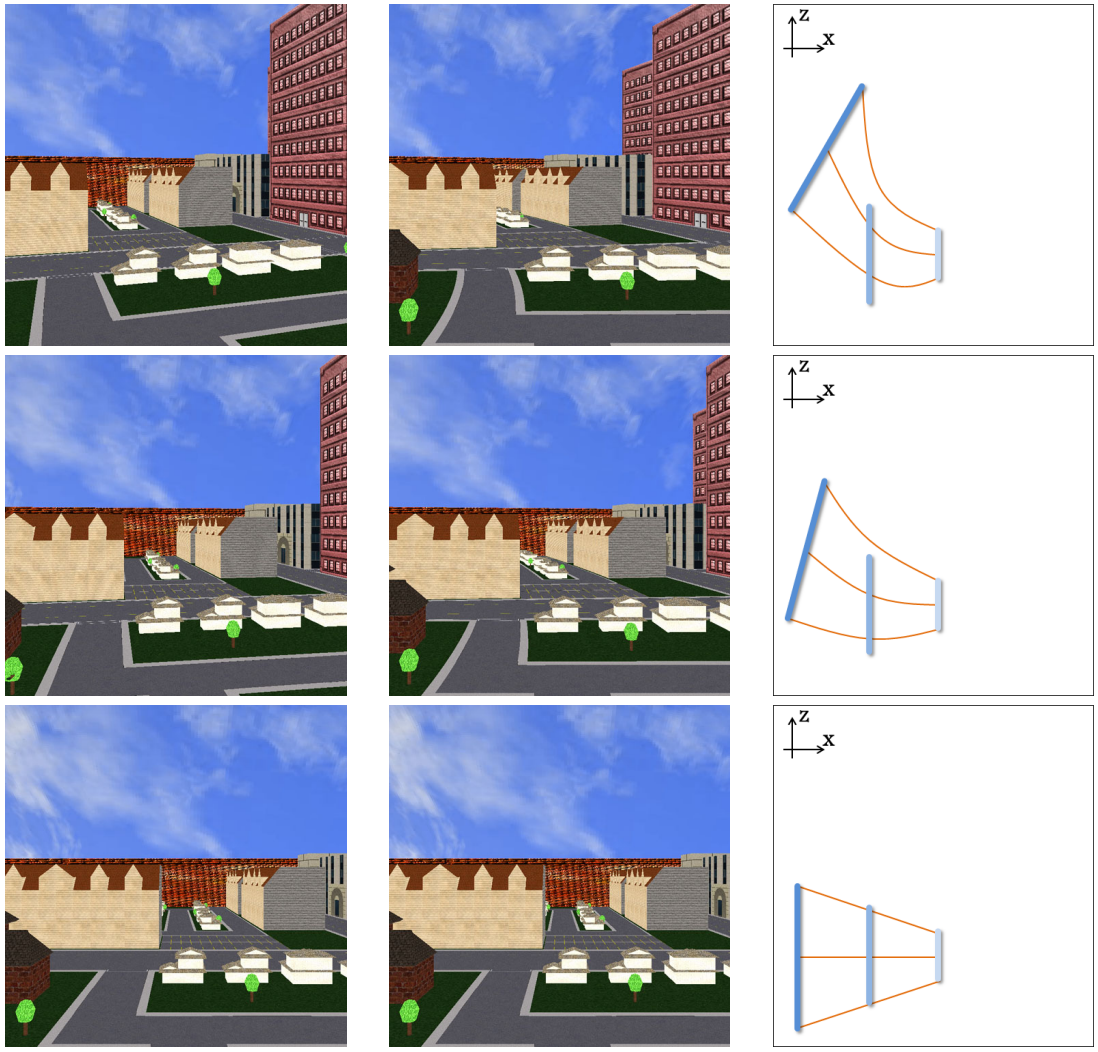


Figure 8.5: Comparison between a camera that uniformly rotates 9° per frame (left) and a staggered camera (center). The images on the right show the top-down positions of the surfaces used to created the staggered camera (3/3).

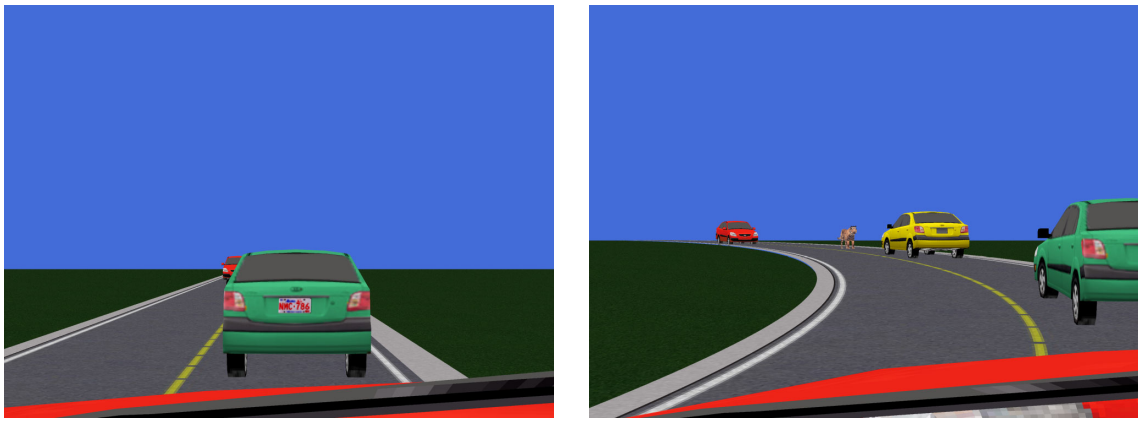


Figure 8.6: Left: the view from a car stuck in traffic. Right: an altered nonlinear projection allows the viewer to see around the cars to the source of the traffic stoppage.

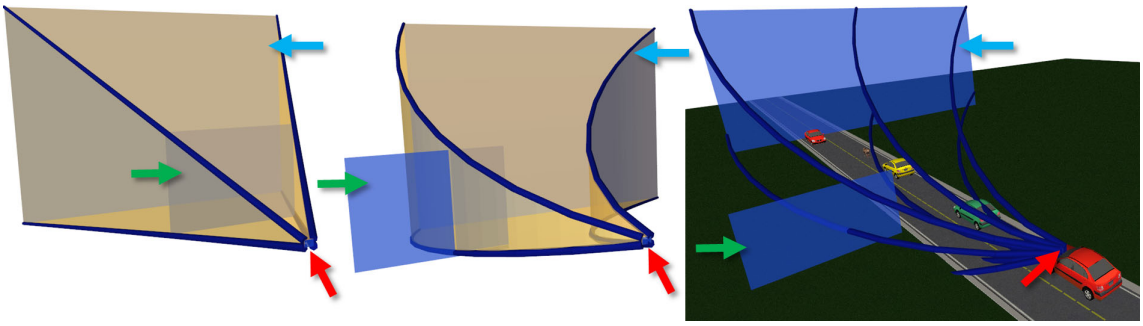


Figure 8.7: The projection volumes used to create Figure 8.6. Left: the starting point, a perspective projection volume created from three planar control surfaces. Center: the end point, a modified projection volume created from moving the middle plane left and the far plane right. Right: the modified projection surfaces and projectors placed within the scene. The red, green, and blue arrows highlight the near, middle, and far control surfaces respectively.

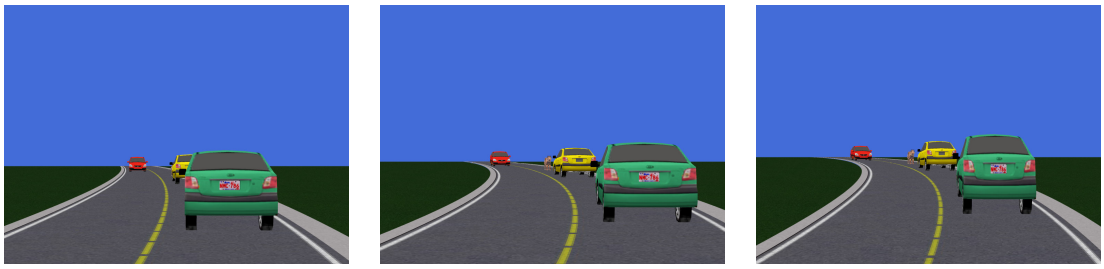


Figure 8.8: Frames created by interpolating between the viewing volumes shown in Figure 8.7.

Figure 8.9: The left image is a perspectively projected view of a saloon. There seems to be somebody standing behind the saloon door but this person cannot be identified. In the right image the projection has been modified to reveal the cowboy.

small opening. Figure 8.9 provides a prototype of such a scenario where the cowboy standing behind the saloon door is revealed by an altered projection.

As in the previous example the initial projection is a three plane perspective projection (Figure 8.10). The middle surface is scaled so that it is only a small amount larger than the gap under the saloon doors. The near and far surfaces are not modified. Instead of the quadratic Bézier curves used previous examples this projection makes use of piecewise linear projectors. This ensures that the middle surface is exactly interpolated, resulting in the projection behaving like the concatenation of two perspective projection volumes: the first being a long, narrow frustum the reaches the saloon doors while the second is a frustum that would normally be associated with a large field-of-view. The projection volumes are shown in Figure 8.10.

In both of these examples one might argue that animated cameras are an overly complex solution to a simple problem where simply moving the camera can provide an alternative solution. The key element of the usefulness of animated cameras in these scenarios is that the viewpoint does not change. This maintains to the position of the viewer so that they do not feel as if they are "flying" or otherwise being moved around. Instead they are able to maintain the context of their current position.

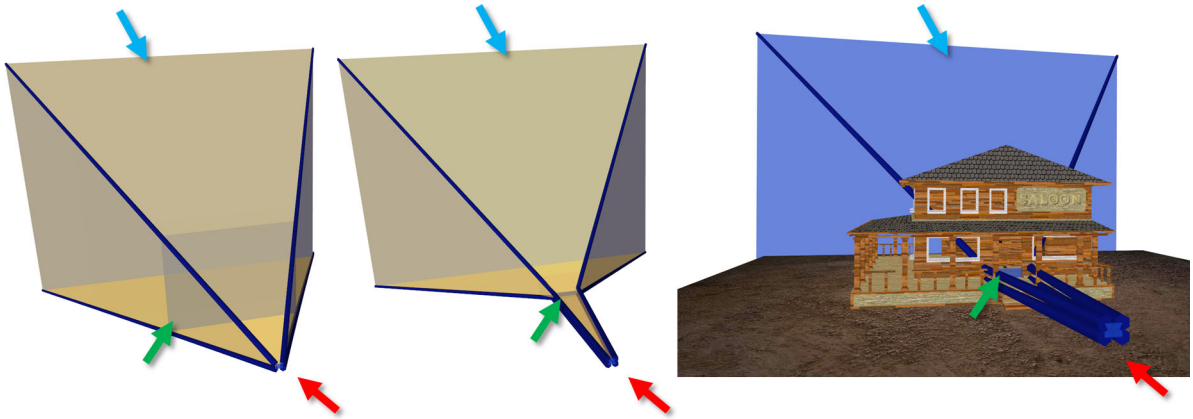


Figure 8.10: The projection volumes used to create Figure 8.9. The left image is of the starting condition, a perspective projection created from three planes. The center image shows the surface configuration of the end condition. Note that the projectors are following piecewise linear paths. As can be seen in the right image, the middle surface was been scaled to barely overlap the bottom of the saloon door so that the cowboy can be seen in the resulting image. The red, green, and blue arrows highlight the near, middle, and far control surfaces respectively.

There exist a wide variety of less trivial examples where avoiding occlusion is useful. One important area is that of 3D data visualization. As mentioned in Section 3.3 Carpendale et al. [CCF97] have addressed occlusion avoidance through displacement. Animated cameras may provide another tool where particular paths can be taken through 3D volumes within the context of the entire volume by manipulating these control surfaces. Another more involved example is the use of such an animated camera within a data volume.

8.1.3 Focus of Attention

Another possibility with animated cameras is to change the projection's characteristics to focus the viewer's attention on specific elements in the scene. An example, shown in Figure 8.11, has the initial condition of a perspective projection directed down a street. Consider the scenario where we would like to now focus the viewer's attention on the red building in the middle of the scene. The emphasis on the red building is created by

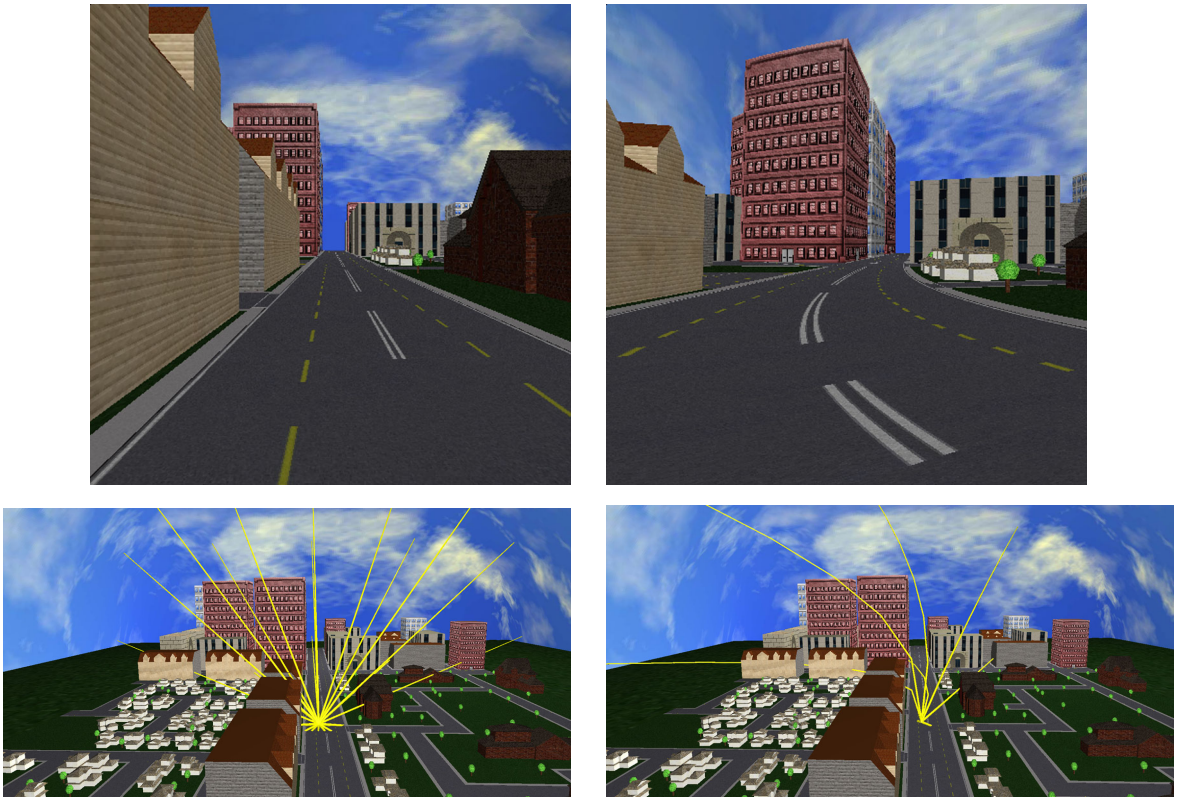


Figure 8.11: Top: Starting with a perspective projection (left) the projection volume is changed to focus on the red building (right). Bottom: nine projectors (yellow) of the perspective projection are shown within the scene geometry.

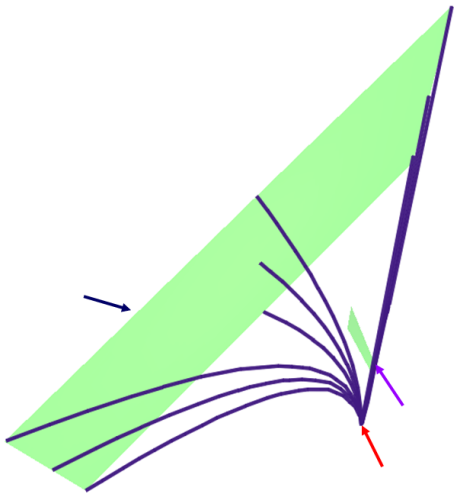


Figure 8.12: The orientation of the three control surfaces used to create altered projection shown in Figure 8.11. The near, middle, and far surfaces are identified by the red, purple, and blue arrows respectively.

repositioning the control surfaces in such a way as to bend the projectors toward the left of the scene, around the buildings in front of the red building, hitting the central building directly. Figure 8.12 shows the orientation of control surfaces used to create this "focused" projection. By creating a perspective projection from three control surfaces we can easily interpolate between the two projection volumes to create as many in-between frames as desired.

A variation on this focus of attention camera is altering a camera that presents a character's first person viewpoint and peripheral view to emphasis a region that is being focused on by the character. Animating the change in projection subtly indicates the character's state and directs the viewer's attention. Figure 8.14 provides an example image where the projection has been modified to emphasize the cowboy in the scene compared to the unfocused example shown in Figure 8.13.

The projection used to create Figure 8.13 is relatively straightforward, created by projection onto a hemicylinder. Figure 8.14's projection is more complex. The goal in this projection is to concentrate more of the image's pixels upon the center of the scene. To achieve this we alter $Q_f(u, v)$ in such a way that will ensure that the center of u's range covers more of the center of the scene than the u parameter in the hemicylinder.

One way to accomplish this change is to use the reparameterization step discussed in Section 4.3. However using reparameterization creates no visual difference in the projection surfaces, making it difficult to link images' characteristics to the projection surfaces.

Instead, for this example, we create a new parametric surface as shown in Figure 8.16. This surface begins with a hemicircle, the top-down view of the hemicylinder. Then this curve is altered by extending the center of the curve outward. This extension shown in Figure 8.15. This curve is then parameterized based on arc-length, linking the shape of the curve to its parameterization. This provides the horizontal extent of the altered

Figure 8.13: Normal state of the peripheral view focusing example. This image is the result projection onto a hemi-cylindrical projection surface (shown in Figure 8.16).

Figure 8.14: Focused state of the peripheral viewing focusing example. The projection used to create this image has horizontally expanded the scene directly in front of the viewer so that it takes up more of the image. This causes the scene to the sides of the viewer to become compressed. The projection surface used to create this image is shown in Figure 8.16.

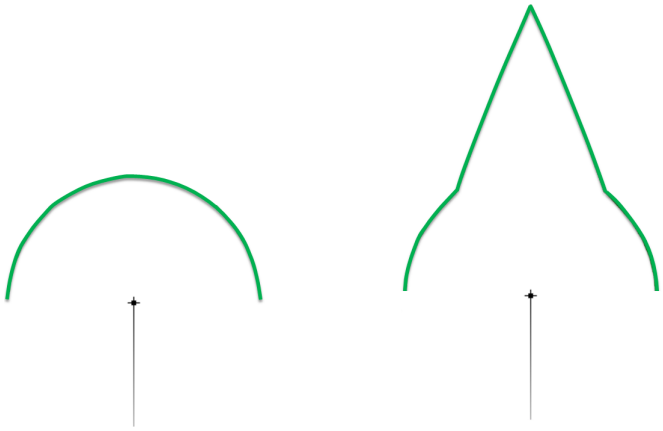


Figure 8.15: The outlines used to create Figures 8.13 (left) and 8.14 (right). The increase in the arc-length at the top of the right outline causes an increase in the sampling of the center of the scene directly in front of the viewing in Figure 8.14.

projection surface. The next step is to extrude the curve upward and downward to create a surface. If we were to extrude the surface to the same height as the hemicircle then the vertical field-of-view will be reduced in the central area where the surface is farther from the viewpoint than the hemicyclinder. To maintain a constant vertical field-of-view we scale the extruded height of the cylinder based on the distance from the viewpoint. Figure 8.16 shows the surfaces used to create the two projections as described.

Another variation is to use these same two projections, but interpolate between them based on the speed the character is moving so that at low speeds the scene is presented uniformly but at faster speeds the view emphasizes objects directly in front of the character while the peripherals become compressed. This could provide a subtle cue that differentiates between speeds where scenery is passing too fast to differentiate as well as to increase the sensation of movement and speed.

8.2 Art Installation Project

While Flexible Projection can be utilized to create specific effects within an image, it can also be used for artistic purposes. Artistically, projection is a means of organizing

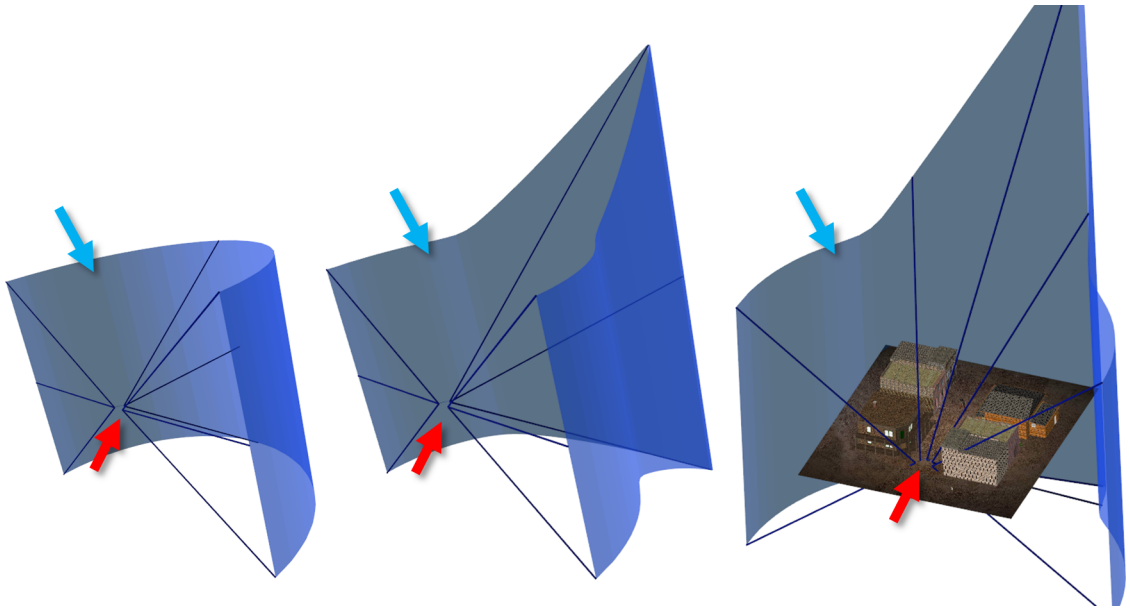


Figure 8.16: The control surfaces and projectors used to create the images shown in Figures 8.13 and 8.14. Left: the surfaces used to create the initial image; the near and far surfaces are hemi-cylinders. Center: the surfaces used to create the second image. While the near surface remains a small hemi-cylinder, the middle of the far surface has been expanded modified to, through its parameterization, concentrate more pixels in the center of the scene. Right: the second projection's surfaces embedded in the scene's geometry. The red and blue arrows identify the near and far surfaces respectively.

space, relationships between objects, and the observer's presence in an image. Flexible Projection provides a tool whereby this organization and presence can be made accessible to artists familiar with 3D surface modeling.

This subsection provides a description of an art installation, *Perspectives*, that made use of Flexible Projection with the aim of exploring the 2008 confrontation between Tibetans and the Chinese authority. This installation was a part of a collaborative course between Hao Wang, a student at the Alberta College of Art and Design (ACAD), and the author of this thesis. In particular, this project portrayed news images from distorted viewpoints representing the biases that exist in the media as well as between the involved parties. The produced images do not just represent pictorial organization but are instead intended to reflect different constructions of this issue and the world in general. [1]

Within the project, Flexible Projection provided a variety of distorted outlooks in realtime. Flexible Projection was particularly useful in creating projections for this purpose in two ways. The first was that these projections could be adjusted to affect different depths of the scene differently; as a result, the background of the scene could feature a different kind of distortion than the foreground. Another aspect was that Flexible Projections can be animated by moving the projection surfaces over time. This added an extra impression of change and distortion. Ultimately Flexible Projection allowed for creation of projections that provided very different outlooks into a virtual world.

As the aim in the project was to produce realtime renderings, we made use of the scanline rendering algorithm discussed in Section 7.3. We made use of two types of control surfaces: spheres parameterized by longitude and latitude as well as parameterized rectangles. As stated in Section 7.3, once the control surfaces have been selected it must

[1] This sub-section remains largely as it was published as a subsection of "Art and nonlinear projection" in the Proceedings of Bridges, 2009 [BCS+09].

be possible to calculate the surface parameters given a point on the surface. For the sphere this can be easily done by converting the Cartesian coordinates to polar coordinates and then shifting and scaling the coordinates to ensure they are within the range $0 \leq u, v \leq 1$. The parametric rectangles were assumed to be parallel to the XY plane and were defined by the top-left P_0, bottom-right P_1 corners, and depth d providing the following parametric definition:

$$Q(u,v) = \begin{cases} (1-u)P_{0_x} + uP_{1_x}, \\ (1-v)P_{0_y} + uP_{1_y}, \qquad u, v \in [0,1]. \\ d \end{cases}$$

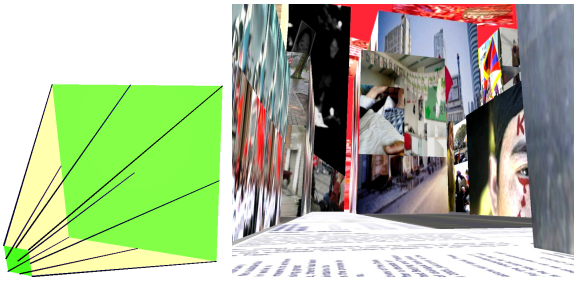
The inverse for use in rendering calculations is

$$Q^{-1}(x,y,z) = \begin{cases} \frac{x-P_{0_x}}{P_{1_x}-P_{0_x}}, \\ \frac{y-P_{0_y}}{P_{1_y}-P_{0_y}} \end{cases}$$

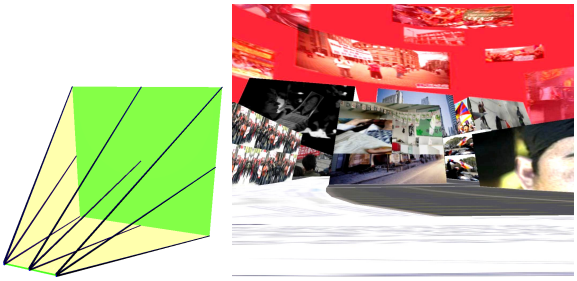
assuming that $z = d$.

Sixteen different projections were developed. The only projection making use of parameterized spheres was a hemispherical projection (also known as fisheye) that was defined by a small hemisphere as the near surface and a larger one as the far surface. The other projections were defined by rectangular patches that, in turn, were defined by their corner points. The simplest projection recreated a perspective projection using two rectangular surfaces is shown in Figure 8.17(a).

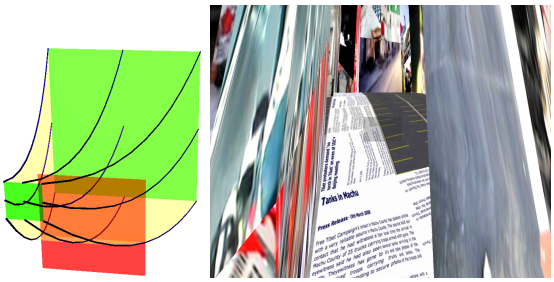
The remaining fourteen projections were essentially variations on perspective projection. This was a deliberate decision, made because it allowed the resulting projections to be different enough to be noticeable and eye-catching, but retain enough familiar characteristics to be intelligible. The modifications made included: swapping the near and far surfaces to create an inverse perspective projection, moving the far surface relative to the near surface to create off-axis perspective projection, making the tops of the near and far surfaces narrower than the bottoms to expand the top of the image, and decreasing the



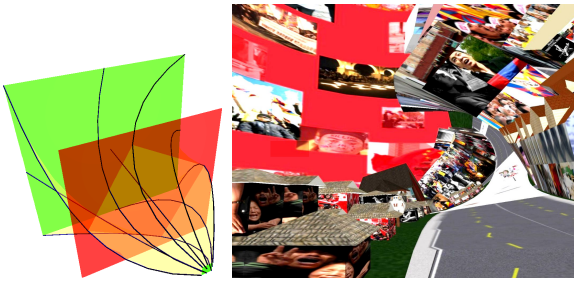
(a) Perspective projection (for comparison).



(b) A projection where the near surface is a line, same view position as (a).



(c) Projection that causes the ground to appear to bulge upwards, same view position as (a).



(d) Twisted projection; causes rotation around the center of projection, view position near to that of Figure 8.18.

Figure 8.17: Projection diagrams (left) and resulting images (right). In the diagrams the near and far surfaces are green, the viewing volume is yellow, and projectors are shown with lines. The red middle surfaces in (c) and (d) are used to curve the volume.

height of the near surface as shown in Figure 8.17(b) to cause stretching of objects close to the near surface. More complex modifications were created by adding an intermediate surface between the near and far surfaces to allow for nonlinear projectors. In one projection, shown in Figure 8.17(c), this intermediate surface has been placed below the near and far surfaces causing projectors to travel downward before curving upward to reach the far surface. This produces an upward bulge in the midground of the produced images. A more exciting result is caused by introducing this intermediate surface and then rotating the far surface by ninety degrees. This, as shown in Figure 8.17(d), results in a twisting through the depth of the image.

Two animated projections were created. The first simply decreased the size of the far plane over time and then increased it to its original size producing an effect similar

to that of zooming in and subsequently zooming out. The second, shown in Figure 8.18, translated an intermediate surface around a circular path.

The project itself made use of a game-like 3d environment with limited interaction. The environment, shown in Figure 8.19, was textured with news stories and images related to recent events in Tibet and related protests in Canada. Two modular ambient display screens [SPC04] were used, each presenting a different virtual camera with a different projection of the environment. Each camera is then translated and rotated along a cyclical path through the environment. The interactive element of the projection was a button that randomly changed the projections used on both screens. Realtime rendering was at a rate of approximately 25 frames per seconds on two display devices of 1280 by 1024 resolution with an Intel Core2 6600 CPU, an NVIDIA 7800 video card, and 2 GB of system RAM. For the installation, the two displays with different projections were placed side by side at eye level. The installation setup is shown in Figure 8.20 and was presented at the ACAD graduation show in May 2008.

8.3 Recreating Artistic Projections

As stated in Chapter 1 a major motivation behind this work is to expand the possibilities inherent in projection beyond perspective and allow elements of nonlinear projection to be controlled in a fashion that would start to allow artistic expression through projection, such as we see in many paintings and other artistic works.

In this section we have taken a well known artistic work, Vincent van Gogh's *The Bedroom*, 1889 (shown in Figure 8.21) and attempted to create a rough reproduction using the non-perspective projection used in this work. The particular painting was chosen due to an in-depth analysis of its pictorial space (and thus projection) by Heelan [Hee72] that concludes that van Gogh's later paintings depict a non-Euclidean space. This non-

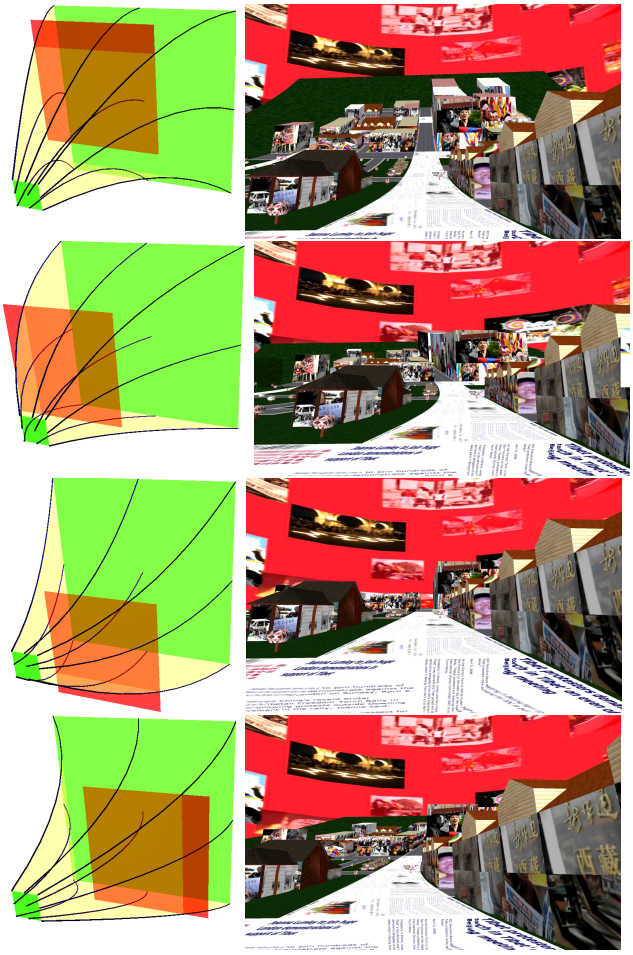


Figure 8.18: Keyframes of an animated projection. The middle rectangle that controls the curving of the volume is moved in circles causing the middle ground of the produced images to bend in a circular motion.

Figure 8.19: The virtual environment used in the project.

Figure 8.20: The *Perspectives* installation.

Euclidean space seems a clear result of a nonlinear projection. Examining Figure 8.21 one can see evidence of this nonlinearity in how parts of the room closest to the viewer seem to spread outward, toward the viewer as evidenced in the shape of the foot of the bed, and the patterns in the strips of the wood floor. Additionally, while the back of the room appears to recede from the viewing as can be seen in the curved line at the bottom of the back wall.

In order to attempt to recreate the painting, a 3D model of the room was necessary. The measurements and relative positions of objects in the room were taken from Hee-

Figure 8.21: Vincent van Gogh, Dutch, 1853-1890, *The Bedroom*, 1889, Oil on canvas, 29 × 36 5/8 in. (73.6 × 92.3 cm), Helen Birch Bartlett Memorial Collection, 1926.417, The Art Institute of Chicago. Photography © The Art Institute of Chicago.

lan's [Hee72] suggestions. To assist in reproducing the likeness of the painting, many of the textures used in the model were taken from the digital copy of the original painting. The resulting model is shown in Figure 8.22.

Figure 8.23 shows two attempts to create the nonlinear projection similar to the one used in the original painting. Both of these projections were created using the point-based interface discussed in Chapter 5. For each of the 5×5 grid of projectors and their image space positions the original painting was examined to determine the appropriate position to intersect each project with the 3D scene (see Figure 8.24). After placing all of the projector's intersection points, the intersection tangents were altered so that the left chair was viewed from from the front of the chair, and less from the side. Tangents of the points intersecting the bed were also changed in order to make the projectors intersect the bed from the side. The points and tangents used in Figure 8.23 were placed individually

Figure 8.22: A 3D model, shown in perspective, that was used to recreate *The Bedroom* in Figure 8.23.

and not reused between the two images. Each of the projections required about three hours to create, this includes the time to specify the points and tangents as well as time to render the image and iterations to adjust points to more closely approximate the painting. The volume used to create the right image of Figure 8.23 is shown in Figure 8.25.

In examining Figure 8.23 it is clear both images only approximate the image; there are several reasons for this. The first is that there are many elements of the 3D model that likely do not exactly reproduce the geometry of the original room. The second is that the artist, while painting, is not constrained to a continuous volume and thus can touch up elements of the nonlinear projection that the artist may not find pleasing. For example, in the Figure 8.23-left the walls intersections lean slightly to the left; this is easy for an artist to fix but more difficult to address in the current Flexible Projection interface.

The interesting aspect of recreating such a projection is the potential to then use this projection to create entirely new images. Figure 8.26 provides the results of applying the created projections to a variety of different scenes and, in one case to a slightly altered

Figure 8.23: Two attempts at reproduction of Van Gogh's *The Bedroom* using the model shown in Figure 8.22. Both images were created separately using the point-based interface. The left image made use of Bézier surfaces while the right image used piecewise linear surfaces.

Figure 8.24: When creating projectors using the point-based interface the painting's image was first consulted to determine where the projector should intersect the scene. In the left image the red lines highlight the image space position corresponding to the projector we wish to place. Left image modified from same source as Fig. 8.21. The right image shows placement of the corresponding projector in the 3D scene. The red circle highlights the specified intersection of the projector (yellow) with the scene.

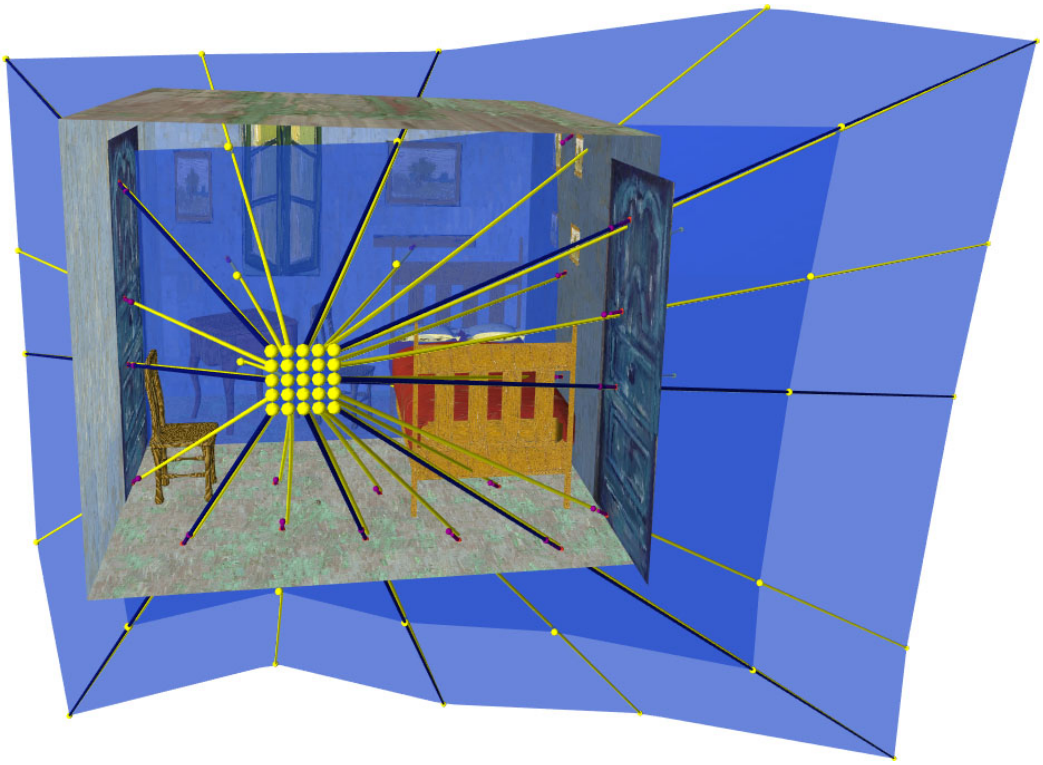


Figure 8.25: The projection volume used to create the right image of Figure 8.23.

version of the Bedroom scene with a slightly different position and angle. While it is difficult to quantify whether the produced images embody the same spatial organization of van Gogh's *The Bedroom*, the resulting images do have interesting appearances that reflect an out-of-the-ordinary arrangement of 3D space in a 2D image.

8.4 Summary

In this chapter we have explored three diverse application of Flexible Projection. The coherent thread between them is that they demonstrate Flexible Projection's utility for a wide variety of purposes and applications. In the next chapter we explore a single application, the creation of full-view customizable panoramas, in-depth and make use of Flexible Projection as the starting point for this examination.

Figure 8.26: Applying the projection based on van Gogh's *The Bedroom* to different/altered scenes.

Chapter 9

Application: Panoramas

This chapter presents an in-depth application of the Flexible Projection Framework in the creation of customizable panoramic projections. Additionally it presents an interface for creating projection surfaces that provides explicit control over the surface parameterization. Surface parameterization is a key element that provides control over the resulting image characteristics.[1]

Figure 9.1: *Panorama of Edinburgh* by Robert Barker, 1792. Image from [Cen07].

Generally a panorama is an image depicting a wide angle view. The word panorama was coined in the late 1700s to describe Robert Barker's paintings on the inside of large cylinders [Com99]. An example of one of these paintings is shown in Figure 9.1. This research focuses on creating full-view panoramas in virtual environments; that is, panoramas with a single view position that encompass the entire 360 degree surroundings. In real-world photography these panoramas can be created by taking several photographs while rotating the camera around a fixed point and then stitching them together into a single image.

Many types of full-view panorama exist; common examples include cylindrical, spherical, and conical panoramas. These panoramas differ from one another both mathematically and in image characteristics. While these descriptions in terms of shape provide

[1]This chapter has been adapted and extended from [BS10], ©2010 IEEE. Reprinted with permission, John Brosz and Faramarz F. Samavati, Shape Defined Panoramas, In Proc. of Shape Modeling International, 2010.

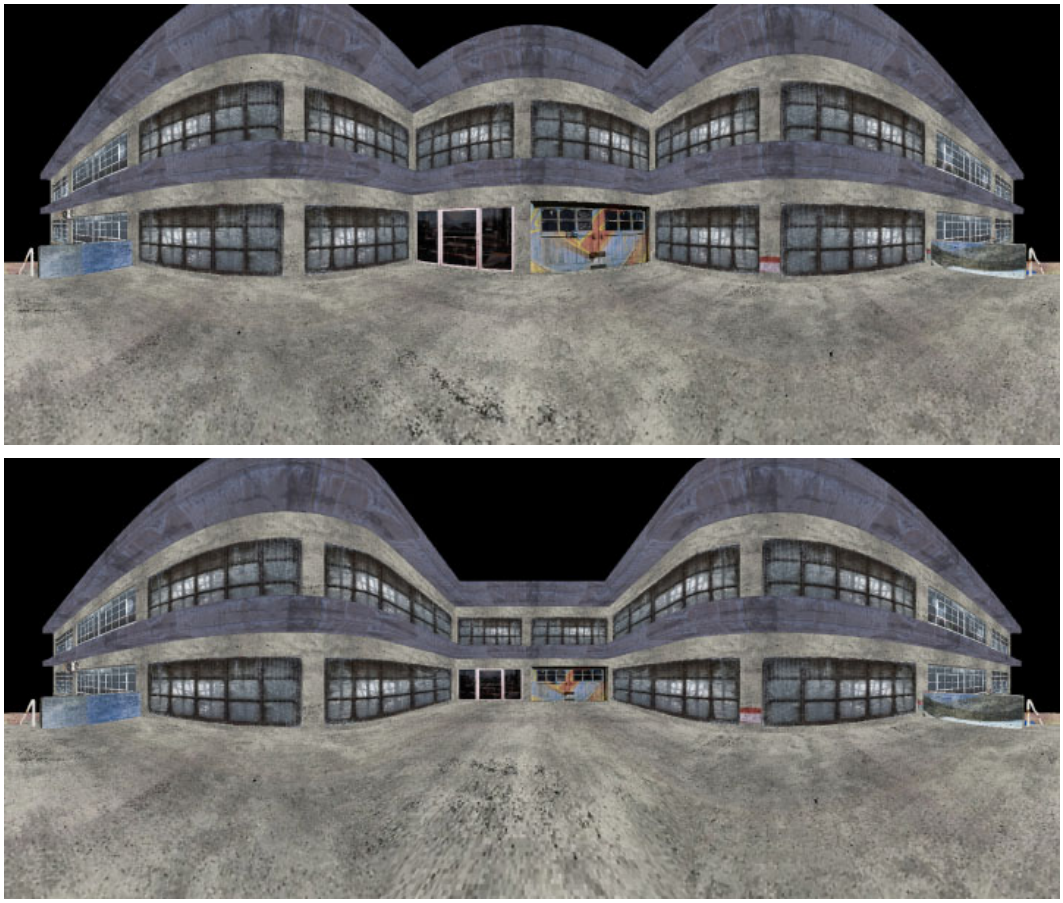


Figure 9.2: A cylindrical panorama (top) is altered to display the center of the image in perspective (bottom).

hints of definition, they are concretely defined through projection equations (i.e., mappings from three dimensions to two). An unfortunate aspect of these equations is that in order to modify them, one must change the surface or its parameterization and re-derive a projection equation, a non-trivial task especially when several iterations are required to create the desired effect. Creating a panorama in this fashion is equivalent to "hard-coding" a scene description; while serviceable, it greatly impairs the ability to explore other possibilities. The approach in this research is to make use of Flexible Projection to bypass this hard-coding.

There are many reasons why these panorama variations may be useful. For instance, a panorama that combines a spherical and a cylindrical panorama reduces the vertical

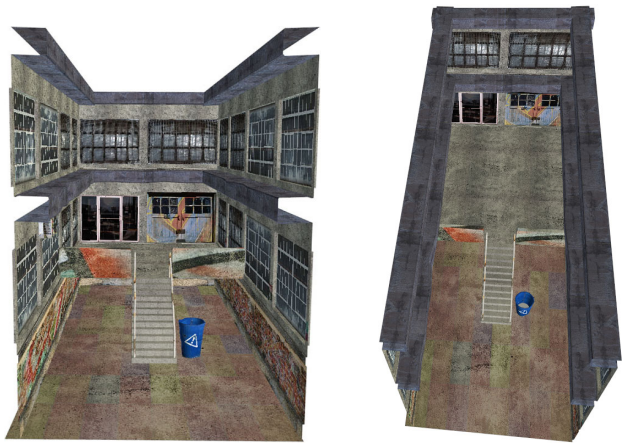


Figure 9.3: Two images of the 3D model used to create Figure 9.2, shown in perspective.

distortion associated with spherical panoramas while capturing the surroundings above and below the viewing position that would be missed by a cylindrical panorama. Another possibility is changing areas of the image to behave as if created by a linear perspective projection, thus avoiding the curving of straight lines produced in most panoramas (as shown in Figure 9.2). Additionally, custom panoramas could be useful for tailoring projections to produce images exactly suited for specific configurations of cave or dome environments. While customized panoramas can be useful for creating panoramas from existing (camera) images, the primary usefulness of this work is to provide new techniques for creating imagery from CG environments.

In this work, panoramas are created by defining a projection surface. This can be any arbitrary surface that encompasses the scene and viewing position. However, the shape of the surface is not enough to define the panorama projection; we also require some specification of how to unwrap and flatten this surface to create a 2D image parameterization. We show that arc-length parameterization can naturally map existing panoramas to their characteristic surfaces (i.e., sphere for spherical panorama and cylinder to cylindrical panorama). Consequently we apply arc-length parameterization to create a unique correspondence between the shape of the projection surface and the characteristics of the

produced image. This correspondence makes it possible to visualize the panoramic projections' behavior by displaying the projection surface, allowing customized panoramas' behavior to be more easily understood and predicted.

Additionally this chapter describes how the Flexible Projection rendering can be adapted for the specific conditions of full-view panoramas. Lastly we describe several panoramas designed for particular purposes and applications.

9.1 Previous Works Related to Panoramas

Beyond the literature reviewed in Chapter 3 there exists a number of works that specifically focus on creating, analyzing, and applying panoramas. This section highlights these additional works and discusses panorama specific properties of a few previously mentioned works.

Salomon provides a good resource for derivations of several panoramic projections including cylindrical, spherical, and conical panoramas.

In applying panoramas Greene [Gre86] describes how projection onto a cube can be used for efficient environment mapping. Glaeser and Gröller [GG99] describe the use of segments of spherical projections to reduce wide angle distortion. Polack et al. [PPC97] perform a study showing that cylindrical projections improve size and depth perception. Szeliski and Shum [SS97] describe one of many methods in which image sequences taken with perspective projection can be stitched together into a single panoramic image. Their technique is noteworthy in that it works without explicit knowledge of camera orientation.

Many works construct multi-viewpoint panoramas; projections constructed by combining images from different viewpoints into a single image. Several of these works have been previously mentioned in Section 3.6. Rademacher and Bishop [Rad99] create single images from long, oriented camera paths for the purpose of replacing image sequences

in image-based rendering. Román et al. [RGL04] and Agarwala et al. [AAC+06] piece together images taken along a roughly planar path. Wood et al. [WFH+97] create multi-perspective panoramas that provide changes in viewing direction, position, and field-of-view within a single image for the purpose of creating backgrounds for cel animation. Yu and McMillan [YM04a] create panoramas modeled after Wood et al.'s [WFH+97] panoramic constructions by combining the output from many different types linear cameras. These cameras must be manually positioned and adjusted to achieve the desired image. Additionally, the camera placement must consider a set of adjacency rules that ensure image continuity. Degener and Klein [DK09] describe a technique where the 3D model is deformed rather than the camera to create appealing images. Agarwala et al. [AAC+06] review many other multi-viewpoint panoramas.

Several works have focused on removing distortions in panoramic or wide-angle images. Zorin and Barr [ZB95] correct photographic, perspective images by adjusting the interplay between perspective and spherical (direct view) projections. Zelnik-Manor et al. [ZMPP05] remove distortion in single-viewpoint panoramas by stitching together either horizontally or vertically neighboring perspective photographs into a single image. Discontinuities at seams are handled by user-assisted placement at feature edges. Due to this technique's ability to only stitch in one direction, it cannot address deformation near the poles of the panoramic projections. Most recently Carroll et al. [CAA09] perform optimization on wide angle images with the guidance of user-provided indications of straight lines. This work is not demonstrated for full-view panoramas and cannot resolve distortions in images that feature parallel lines that converge at on-image vanishing points [CAA09].

As mentioned, the techniques in this chapter are constrained to single-viewpoint panoramas that encompass the viewpoint's entire 360 degree surroundings. The most related work to this is that of Trapp and Döllner's [TD08] single-center projection. In

their work, that includes but is not limited to panoramas, projections are created by specifying a normal map. The normal map controls the projection, specifying the direction from the center of projection where objects will be projected. Rendering is accomplished in realtime by creating a cube environment map that is resampled with hardware in a second rendering pass to create the projected image.

In comparison, the technique this chapter presents has several strengths. The first is that we describe an object space-based rendering technique that avoids image sampling issues. The second is that it uses surface modeling rather than normal maps to specify our panoramic projections; this allows the use of existing modeling tools to ease the task of specifying panoramic projections.

9.2 Panoramas

Panoramas can be described as projections onto cylinders, spheres, cubes, or other surfaces that surround a viewing point. This viewing point, also known as the center of projection, is a point at which we might imagine the viewer's eye to be positioned. The up axis is the axis around which the viewing direction is rotated.

The process of creating a panorama can be broken into two steps: projection through the eye onto some sort of projection surface; then mapping that surface to a flat, usually rectangular, image. This mapping relies on a 2D parameterization of the projection surface onto a rectangle; creating a seam by splitting the surface and flattening it. A difficulty is that many such mappings can exist depending on the 2D parameterization.

The intention of this work is to develop a geometric technique for creating new panoramas that builds on the specification of existing panoramas. To provide the reasoning behind the chosen technique we begin by examining common types of panoramas: cylindrical, spherical, and cubic.

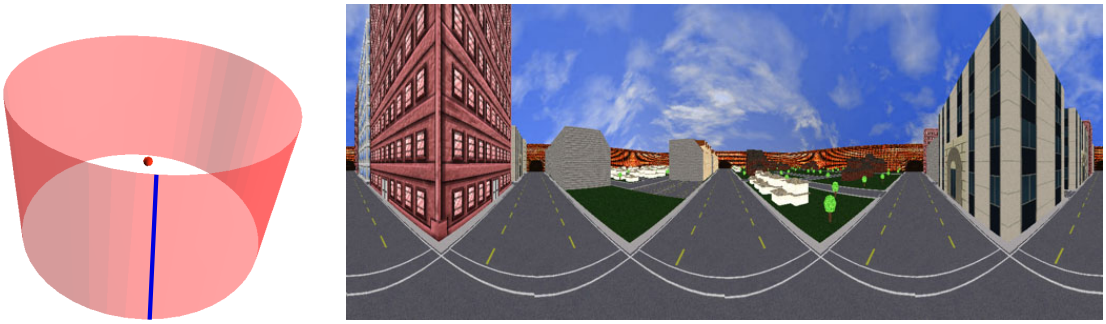


Figure 9.4: A cylindrical panoramic projection surface with marked seam and center of projection (left) and projected image (right).

Figure 9.5: Perspective views of the city model used to demonstrate the panoramas.

A *cylindrical panorama* is most often the result of projection onto an open-ended cylinder; its center coincident with the eye position and its primary axis aligned with the panorama's up axis (Figure 9.4). The original model is shown in perspective in Figure 9.5. After projection onto the cylinder, a simple parameterization is created by cutting the cylinder's surface to form a seam and then unrolling to form a flat, rectangular image.

Let us consider this projection as if we were ray tracing it. When tracing a row of pixels, each pixel's ray will be the same as the previous pixel's, but rotated by some constant angle around the up axis. This is due to the one-to-one mapping between the pixels and points spaced equally around a circle on the surface of the cylinder. For a column of pixels the situation is different since they sample a line segment parallel to the

cylinder's primary axis. Each pixel is spaced equidistant along the line segment causing the angle between samples to vary as shown in Figure 9.6. This angle will be largest as rays near to perpendicular with the up-axis and smaller as they approach parallel. Let us refer to the angle between pixels' rays (or samples) for the rows of the image as θ and the angle between pixels of the columns as ϕ. So, for this type of cylindrical panorama, θ experiences a constant change while the change to ϕ varies.

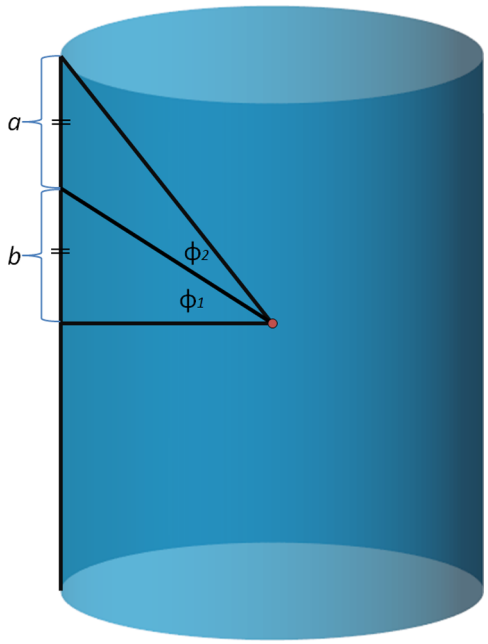


Figure 9.6: When sampling the image columns of a cylindrical panorama, samples are taken parallel to the cylinder's axis. Equally spaced samples along the cylinder wall causes the angle ϕ between samples to vary. In this example the angles $\phi_1 \neq \phi_2$ despite $a = b$.

A *spherical projection* is created by projecting objects onto a sphere. Flattening the sphere creates trade-offs between preserving area, straight lines, angle, scale, and shape [Sny93]. For simplicity we discuss the simplest approach, i.e., an equidirectional spherical projection. This uses the sphere's longitudinal and latitudinal coordinates directly as x and y coordinates in the image as demonstrated in Figure 9.7.

Now let us examine this projection in terms of θ and ϕ. As in the cylindrical panorama θ experiences constant change between neighboring row pixels. The change to ϕ in this

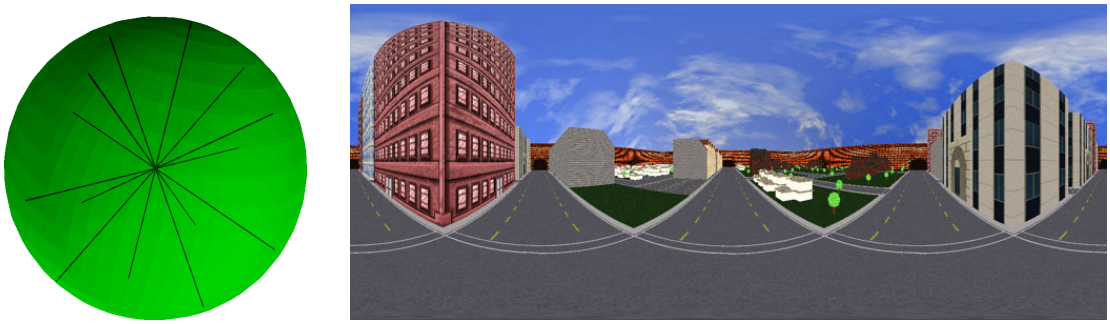


Figure 9.7: A spherical panoramic projection surface (left) and projected image (right). The blue lines within the surface indicate rays that might be used to trace a column of the image.

Figure 9.8: A cubic panoramic projection surface with marked seam (left) and projected image (right).

scenario is also constant since each pair of pixels will be separated by a constant arc-length.

The last panorama we review is a *cubic panorama* created by projecting onto a cube with an open top and bottom. In essence, this panorama is a concatenation of four separate perspective projections (each rotated 90 degrees around the up axis from its neighbor). A cubic panorama is shown in Figure 9.8.

Examining cubic panoramas in terms of θ and ϕ we find that both angles change variably due to equidistant steps along the planar sides of the cube. The variation in angle is largest at the center of each face and smallest at the cube's edges.

By setting the properties of each of the described surfaces we can derive projection equations (see Salomon [Sal06] for an example) and then use these equations to create

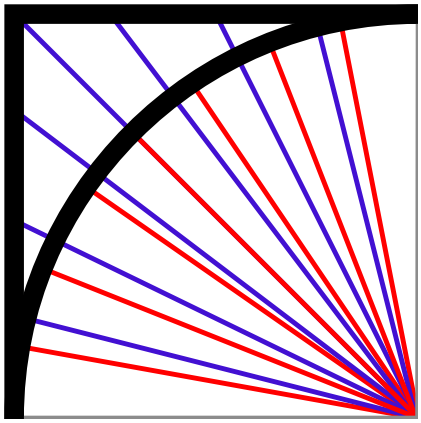


Figure 9.9: Comparison of sampling of a quarter square (blue) and a circle (red), analogous to a sphere and cube.

panoramic images from virtual scenes. While this derivation is straightforward it becomes markedly more difficult if one wishes to experiment or alter these projections. For instance, what if we desire to place the center of projection away from the cylinder's center? It would be convenient if we could automate this derivation process for a variety of shapes and possibilities.

In Figure 9.9 we compare the θ sampling of a cylindrical to a cubic panorama. If we parameterized a cylinder such that the θ sampling matched that of the cube's samples projected onto a cylinder we could exactly reproduce a cubic panorama with a cylindrical projection surface. This makes the projection surface unimportant in determining the projection, since changing the surface parameterization is alone sufficient to control the projection.

So if the surface does not matter, then why use a surface? Working with and changing parameterization to control projections is not an intuitive operation. It is akin to using a hard coded scene description rather than an externally controlled scene description that can be visualized and easily modified. Our approach makes use of the ability of surfaces to be similarly visualized and that they are already associated with conventional panoramic projections. Consequently our goal is to create a technique for specifying panoramic

projection surfaces that uses a fixed correspondence between parameterization and the surface shape.

Cylindrical, spherical, and cubic panoramas introduce important differences in image characteristics. In general panoramas feature trade-offs between spherical projection and planar projections. This is exactly the observation made about projections in general by Zorin and Barr [ZB95]. Spherical projection provides direct viewing, making all parts of the image appear as if they are viewed straight on. This matches the way we rotate our eyes to examine our surroundings. The drawback is that, while isolated areas of the image appear correct, when the entire image is viewed as a whole the result seems strange. These image characteristics are produced by constant angle sampling in the image (e.g., in spherical panoramas and in the sampling of θ in cylindrical panoramas).

Planar projection maintains a global coherence, preserving straight lines. Unfortunately, this projection introduces distortions as one samples away from the center of projection, leading to problems with wide angle images, as well as in projections of circles and spheres [ZB95]. One previously mentioned example of this distortion is shown in the left image of Figure 4.4. Another example is shown in Figure 9.10 where a comparison between a narrow and extremely wide field-of-view highlight the distortions created by perspective projections. These characteristics are seen in cubic panoramas and in the sampling of ϕ in cylindrical panoramas.

In deciding upon a particular panorama one selects a particular balance between spherical and planar projections; the aim of this work is to provide more freedom in adjusting this balance to create desired image properties.

Figure 9.10: An example of distortion in a perspectively projected image. From left to right we extend the a perspective projection from a narrow to an extremely wide field--of-view ($> 160°$). All images have been captured from the same position and direction. The objects near the edges of the widest angle image are very noticeably distorted.

9.3 Controllable Panoramas

The discussion thus far has led to the conclusion that in order to control the composition of panoramas one must have control over the sampling of θ and ϕ (i.e., the spacing between the projection rays used to create the image). Two mechanisms control this sampling: the projection surface and the parameterization of this surface. Due to overlap between these mechanisms, it seems useful to constrain the parameterization so that it is entirely dependent on shape of the projection surface. This will mean that control over the panoramic image can be entirely achieved by control over the projection surface; this has several benefits: modeling 3D surfaces is a common task with a concrete physical analogy and, in general, easier to perform than specifying or modifying a surface parameterization; a 3D surface can be easily visualized providing clear feedback to the user; and common panoramas are currently described based on their projection surface, by continuing to define panoramas by surface shape builds upon existing practices.

The remaining question is how to bind the parameterization to the surface shape. Since the goal is to control the sampling of θ and ϕ it is logical to make use of a parametric surface $Q(u, v)$ and assign one parameter to θ and the other to ϕ. Then, to control the

sampling, we construct the surface from iso-curves that define the sampling. In order to attach the sampling behavior to both curves, we space our samples at uniform distances along the curve, thus parameterizing the curve and eventually the surface by arc-length. Thus a straight curve (or segment of curve) will produce a variable change in sampling (as in the cubic panorama) while a circular arc will produce a constant rate of sampling. This arc-length parameterization can easily be implemented by following Farin's suggestion [Far01] of using chord length to closely approximate arc-length and has the added benefit of allowing use of a wide-variety of analytical curves (e.g., parametric, implicit) as well as discrete ones (e.g., user-sketched curves).

9.3.1 Projection Surface

Thus far the approach is to control the sampling of angles through arc-length parameterized curves. This section first introduces the curves used to control the sampling of θ and ϕ; afterwards the construction of the surface will be discussed.

The sampling of θ is described by a 2D closed curve $P_o(u)$ parameterized by arc-length. This curve is the *outline* and it controls how sampling changes horizontally across the projection surface and the image). If we imagine a row of x pixels in the desired panoramic image and the total arc-length of the outline is L, then each subsequent pixel is $\delta L = \frac{1}{x-1} L$ further along the outline than the previous pixel. Areas with high arc-length will occupy more pixels of the image than areas with lesser arc-length. A circle as an outline produces a constant change of θ as seen in cylindrical and spherical panoramas. The cubic panorama has a square outline. Straight areas of the outline cause sampling of θ to assume the characteristics of planar projection. The corners of the square are treated as vertices of a piecewise curve.

The easiest way to create a surface from the outline is to extrude the outline along the up axis. However since we also want to have control over sampling of ϕ this is not

sufficient. For control over ϕ sampling the user creates a *profile*. A profile $P_p(v)$ is a 2D open curve parameterized by arc-length that controls sampling of the columns of the image. For example, the profile of a spherical panorama would be a hemi-circle whereas the profiles of cylindrical and cubic panoramas are vertical line segments parallel to the up axis.

To create a surface from both a profile and an outline let us first consider a circle as our outline. In this case one can use a surface of revolution (see Farin [Far01] for a formal description). That is, a surface is created by revolving the profile 360 degrees around the up axis. One could also think of this surface as an extrusion of a circle with varying scale along the up axis controlled by the profile's distance from the axis.

To move beyond circles, creating surfaces from other outlines, we use the same construction of extrusion using the given outline rather than a circle. The equation for this surface, assuming the outline and profile have a domain of $[0, 1]$ and have both been defined on the XY plane, is:

$$Q_d(u, v) = \begin{pmatrix} P_o(u)_x P_p(v)_x, \\ P_p(v)_y, \\ P_o(u)_y P_p(v)_x \end{pmatrix}. \tag{9.1}$$

This surface bears similarity to the cross section oversketch surface described by Cherlin et al [CSSJ05]. It should be clear that when the outline is a circle i.e., $P_o(u) = (\sin(2\pi u), \cos(2\pi u))$, that the produced surface is exactly a surface of revolution. Other values for $P_o(u)$ cause the profile to be scaled inward and outward from the origin as shown in Figure 9.11.

In examining the surface created in Figure 9.11 we see that outline directly controls the shape of the surface as seen from above. By taking any slice of the surface perpendicular to the up axis one obtains a scaled copy of the outline. By slicing the surface with a

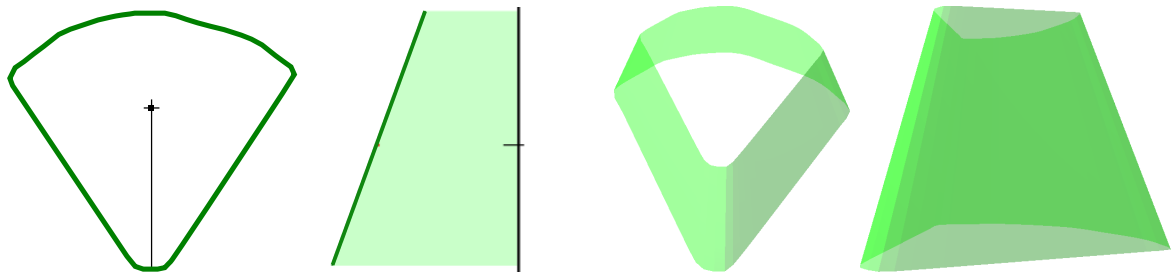


Figure 9.11: An arc-length parameterized projection surface created with Equation 9.1. In diagrams of the outline the origin is marked; the descending line's intersection with the outline indicates the seam. The front of the projection (the center of the image) is halfway along the arc-length of the entire outline from the seam. Profile diagrams show the profile as well as a light fill to relate the curve to the up axis.

plane containing the up axis, one obtains two copies of the profile, one a mirror of the other.

With the surface $Q_d(u, v)$ we can evaluate θ by fixing v, allowing u to vary uniformly, and examining the angular change between the points. As the x and z coordinates of the surface are determined by the orientation of ray around the eye position, it should be clear that the outline $P_o(u)$ provides direct control over θ as the uniform scaling by $P_p(v)$ does not alter the angular change. When examining ϕ and fixing u, the resulting curve is a non-uniformly scaled version of the profile.

To allow greater control over θ and ϕ, it useful to allow for multiple profiles. These n profiles, $p_0, p_1, ..., p_{n-1}$, are associated with the u values, $u_0, ..., u_{n-1}$ denoting where each profile is positioned along the outline. The profiles are ordered such that $u_0 < u_1 < ... < u_{n-1}$. For a given u, either the value is exactly that of one of the specified profiles (in which case that exact profile is used), or it is not. If not, the surface uses linear interpolation between the profiles p_i and p_{i+1} where $u_i < u < u_{i+1}$. This interpolation is based on the parameter $t = (u - u_i)/(u_{i+1} - u_1)$ thus our profile is $p = t * p_i + (1 - t)p_{i+1}$. For end conditions, $u < u_0$ or $u > u_{n+1}$ the calculation interpolates between p_{n+1} and p_0.

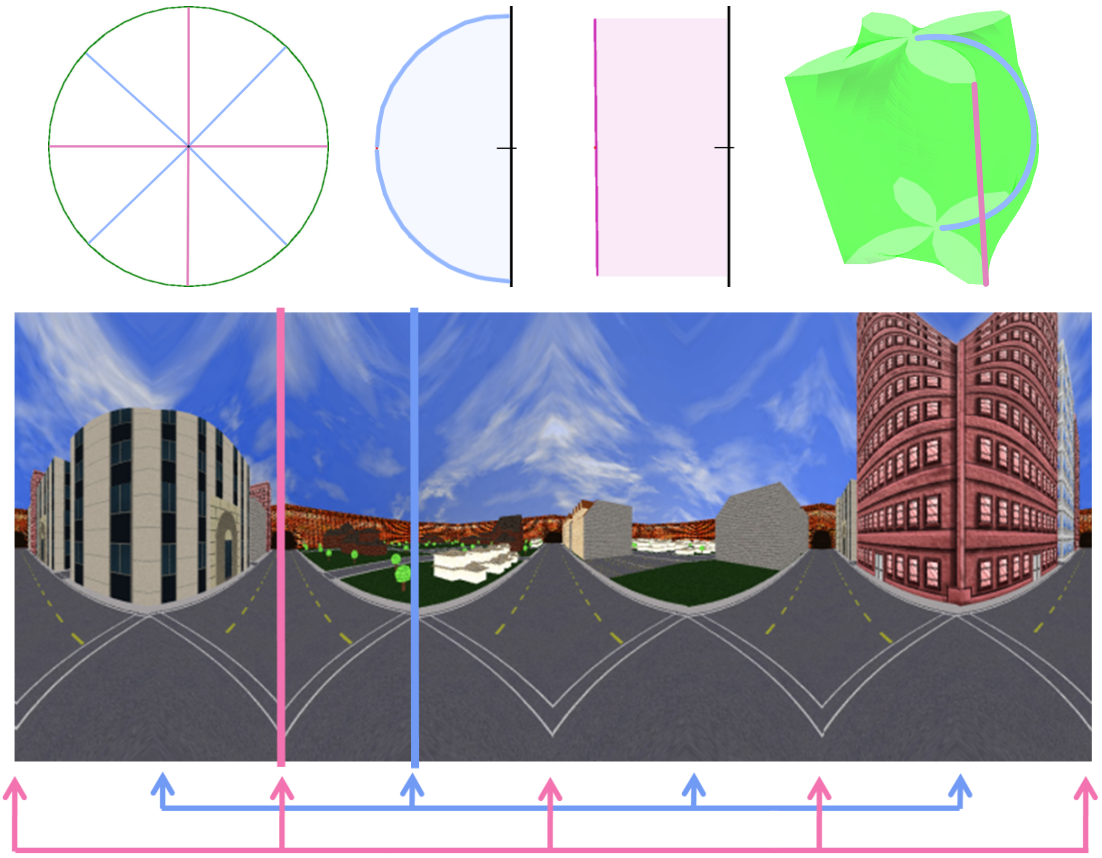


Figure 9.12: An example of multiple profiles blended with linear interpolation. In the top row the images show the outline (left), the profiles (center-left and center-right), the surface (right). The coloured lines on the surface indicate sample positions where the similarly coloured profile has lone influence. The bottom image shows the result of the projection. The two coloured lines indicate two positions in the image where the similarly coloured profile has lone influence. The coloured arrows underneath highlight all of the endpoints of the interpolation.

Figure 9.12 provides an example of a projection surface created with multiple profiles as well as the resulting image.

While linear interpolation is adequate, often one prefers the user-defined profiles to have more influence than merely serving as end points of interpolation. For example, if one profile is a vertical line and the next a hemi-circle then the majority of the image between these profiles will appear as if projected by a flattened hemi-circle due to this interpolation. Instead it would be useful to have most of the area closer to vertical

profile show characteristics of projection onto a plane and most of the areas near the hemi-circle profile appear as if projected onto a sphere with much smaller area of the image showing the effects of the flattened hemi-circle interpolation. This *ease-in, ease-out* effect between specified profiles seems very desirable. This effect can be achieved using a 1D Bézier based interpolation of $p = ((1-t)^3 + t(1-t)^2)p_i + ((1-t)t^2 + t^3)p_{i+1}$, although other interpolation techniques would serve equally well. This *ease-in, ease-out* causes the profiles to remain similar to the defined profiles over a larger surface area and then transition quickly. Figure 9.13 demonstrates both techniques.

In examining the resulting surface one can extract the 2D profile by fixing u. This means that this exact profile creates a column of pixels in the projected image. It is also the case that this curve exists at a particular spherical angle θ. This allows a profile to be easily aligned with a world object. While not immediately important, this is noteworthy later in discussions of creating surfaces from profiles and multiple outlines.

Similarly a surface can be created from multiple outlines. If we desire multiple outlines and a single profile we can use the m outlines $q_0, ..., q_{m-1}$ exactly as we did the multiple profiles by specifying the profile points at which each outline is attached. Because there is no wrapping at the edges we assume q_0 extends fully to the bottom and q_{m-1} extends fully to the top. Elsewhere the surface is created by interpolating between the outlines.

In order to create a surface from both multiple profiles and multiple outlines we must choose how to attach profiles and outlines to one another in order to form the surface. If we choose to again use arc-length parameter values (u and v) we preserve the effect of outlines corresponding to rows in the image and profiles to columns. The alternative approach taken in this research is to attach the profile and outline curves to specified spherical coordinates (θ and ϕ). This causes outlines and profiles to curve through multiple rows/columns in the image but ensures that the profile and outline curves appear exactly horizontal for outlines or vertical for profiles within the surface.

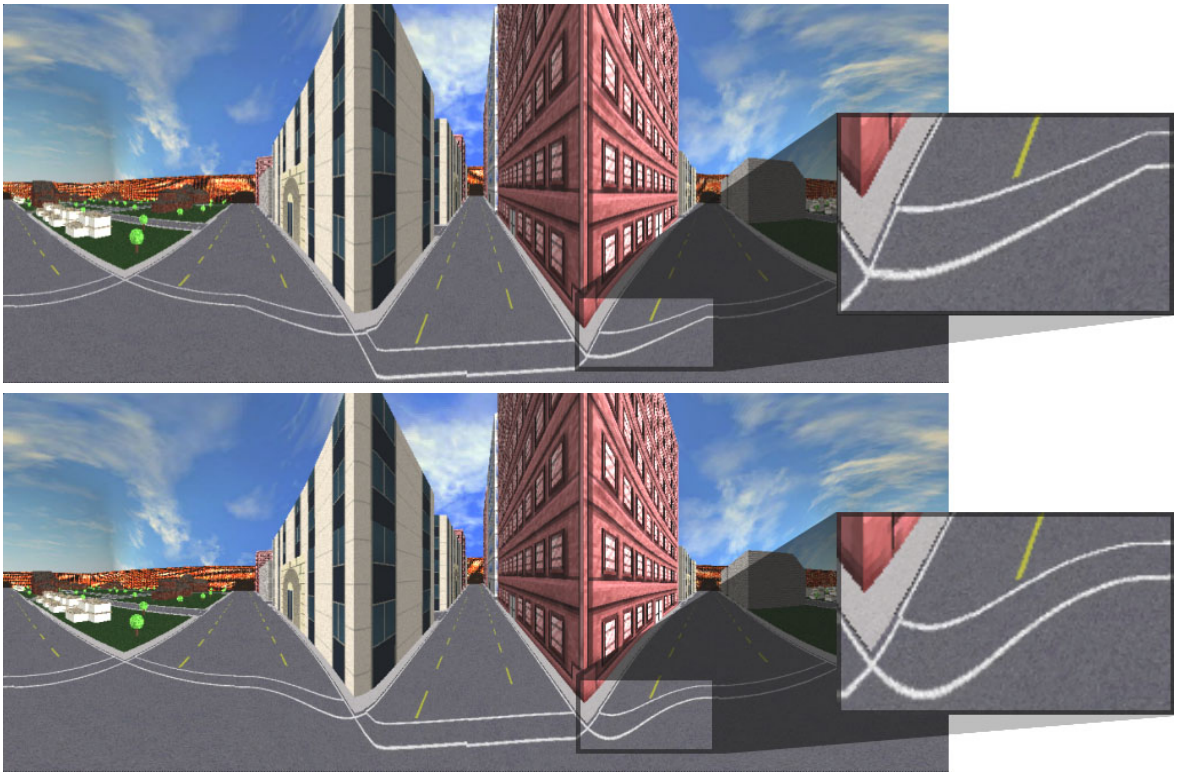


Figure 9.13: Comparison between linear (top) and Bézier based (bottom) interpolation between profiles. Transitions between profiles are most noticeable in the magnified area examining the crosswalk lines. With linear interpolation the crosswalk lines transition sharply; Bézier interpolation creates a curved transition.

This is useful as it means there is a direct link between outline and profile curves and objects in world space. A drawback of this approach is that it limits our surfaces to those that map one-to-one with spherical coordinates. This is not a large drawback; such situations cause parts of the scene to appear in more than one location in the resulting image; an undesirable scenario in most applications.

With this design choice made, each profile $p_0, ..., p_{n-1}$ is positioned by a spherical coordinate $\theta_0, ..., \theta_{n-1}$ and each outline $q_0, ..., q_{m-1}$ is positioned by $\phi_0, ..., \phi_{m-1}$. Interpolation between specified curves is performed as in previous scenarios.

Our panoramic surface equation for multiple outlines and profiles becomes:

$$Q_d(u,v) = \begin{pmatrix} P_{o_{f(\theta)}}(u)_x P_{p_{g(\phi)}}(v)_x, \\ P_{p_{g(\phi)}}(v)_y, \\ P_{o_{f(\theta)}}(u)_y P_{p_{g(\phi)}}(v)_x \end{pmatrix} \tag{9.2}$$

where f and g are the interpolation functions used to determine the profile and outline curves respectively thus $P_{o_{f(\theta)}}$ is the outline interpolated from specified outlines based on the spherical coordinate θ.

It is important to note that our panoramic projection surfaces are constrained by the defining outline(s) and profile(s) so that some enclosing surface shapes are not possible (for instance one cannot create a twisting around the camera's up axis). This is a design feature that emerged out of experimentation and finding, as also noted by Carroll et al. [CAA09], that panoramas are often oriented so that the up vector is parallel to vertical lines in the scene and that such lines should remain vertical in the resulting image. Profiles, and their attachment to specific θ values ensure these vertical lines; similarly outlines maintain the horizontal orientation of lines orthogonal to the up axis.

9.3.2 Outline and Profile Behavior

When manipulating outline and profile curves there are two major operations that produce specific image outcomes. The first is making a section of (or the entire) curve a straight line. This creates an area of planar projection. For true planar projection it is necessary to have straight lines in both outline and the profile to produce a truly flat area on the surface.

The second operation is altering the curve to change its arc-length. Reducing the arc-length causes fewer samples to be taken over that area of the image. This is useful if you wish to compress an area of the image. Increasing the arc-length of the shape causes more samples to be taken over the corresponding area of the image and expands

an area's resolution. An example of increasing an area's resolution through changing the outline was demonstrated in the last chapter in Figures 8.13 and 8.14. The outlines used to create these images are shown in Figure 8.15.

9.4 Rendering Panoramas

An important consideration in rendering is that all projector rays originate at the eye and are then directed outward at the entire 360 degree environment. Consequently the specification of the different panoramas only affects the distribution of these rays; visibility, lighting, and other phenomena do not change with the alterations to the panorama specification. Thus it is possible to render a spherical projection of the environment and then use two-pass rendering with image resampling operations to achieve the desired panorama. Trapp and Döllner [TD08] use a similar technique that uses a cube map rather than spherical projection to specify ray direction. While this technique is applicable to panoramic projection surfaces described in the previous section, the use of image resampling in this technique may be problematic due to changes in resolution and aliasing issues.

The fact that controllable panorama projection surfaces are defined in object space provides additional rendering options that avoid the problems from resampling images. Ray tracing, a slower process, uses the arc-length parameterized surface to define ray direction just as with Flexible Projection in general and without the complication of nonlinear projectors. The other alternative is rasterization. However in order to rasterize these full-view panoramas based on specification by outline(s) and profile(s) there are a few important modifications that must be made to the technique for Flexible Projection previously described in Section 7.3.

9.4.1 Ray Tracing

Ray tracing these panoramas draws exactly upon Flexible Projection where the projection's viewing volume is defined by a linear interpolation between a point (the eye) and the arc-length panoramic projection surface. This gives us a parameterized viewing volume $Q(u, v, t)$ where t linearly parameterizes the depth of the projection, and u, v correspond to the parameters of Q_d. If we wish to adjust the distance of the near and far surfaces from the eye position we adjust the range of t. Rays are created by casting a ray from the center of projection through the surface at fixed u, v. To ensure the sampling specified by the projection surface is propagated to the image, we should iterate through the u and v with equal step sizes; for instance pixel (i, j) should be traced with the ray $Q(t) = Q(i/width, j/height, t)$.

While this technique is slower than the other techniques, it is important to note that it is extremely generic, as we are only describing how rays should be positioned and oriented. Due to the fact that these projections do not require nonlinear projectors, these panoramas can be ray traced in any ray tracing implementation from the most efficient realtime ray casting techniques to the most detailed and accurate ray tracing approaches.

9.4.2 Rasterization

The rasterization approach requires development of a projection equation that allows us to project triangle vertices based on the panoramic surface Q_d. The nonlinear projection can then be performed on the GPU with vertex or geometry shaders [AMHH08] by simply substituting our nonlinear projection equation for the standard projection matrix. The remaining work of rasterization (i.e., filling, clipping, etc) is completed by the hardware's default algorithms. Unfortunately, these algorithms assume the use of linear projection, causing two issues that must be dealt with: seams and accuracy. The remainder of this

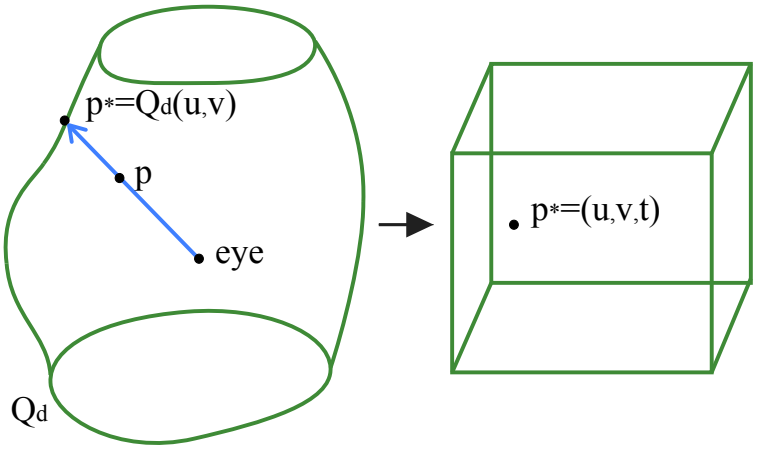


Figure 9.14: When rasterizing a vertex p is transformed from camera coordinates (x, y, z) to parameterized volume coordinates (u, v, t). We can find (u, v) quickly by identifying the ray (with unique spherical coordinates) that passes through the eye, p, and p^* as shown on the left. The right image shows the transformed point $p*$ rescaled into the volume parameters (u, v, t) that are interpreted as normalized device coordinates within the viewbox.

subsection discusses the projection equation, handling seams, and the accuracy of the filled triangles.

Projection Equation. Our main task in rasterization is to develop a projection equation that transforms a point in camera coordinates (x, y, z) to a projected point on Q_d and from there to normalized device coordinates. As before, the projection volume is defined by bounding it between the eye position and Q_d:

$$Q(u, v, t) = (1 - t)(0, 0, 0) + tQ_d(u, v) \quad 0 \leq u, v, t \leq 1.$$

Near and far depth clipping can be adjusted by altering the range of t. The inverse $Q^{-1}(x, y, z) = (u, v, t)$ is our desired projection equation. With Q^{-1} we can rescale (u, v, t) to provide normalized device coordinates. This inverse equation is nonlinear and, depending upon the profile(s) and outline(s), can be difficult to analytically derive.

To simplify the inverse calculation possible surfaces are limited to those that map onto $2D$ spherical coordinates. While this does introduce a restriction, it is only necessary for rasterization and ensures that spherical coordinates uniquely map to points on Q_d;

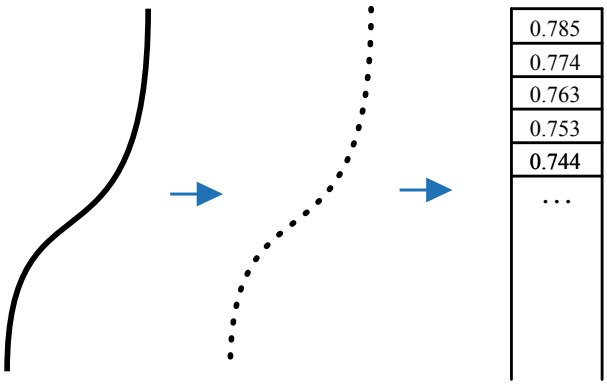


Figure 9.15: Precalculation for outline and profile curves. The curve (left) is first discretized into points based on arc-length (middle) and then the appropriate spherical coordinate (θ for outlines, ϕ for profiles) is stored in an ordered array (right) and transferred to the GPU.

allowing us to use spherical coordinates as an intermediate step in conversion between (x, y, z) and (u, v, t). With the spherical coordinates of p, we can search for the point on the surface Q_d that has the same spherical coordinates. As shown in Figure 9.14 these coordinates uniquely identify the ray passing through the eye, p, and p^* on Q_d. Knowing $p^* = (x^*, y^*, z^*)$ on Q_d and the associated parameters (u^*, v^*) we only have to calculate depth. Depth is the ratio between distance of the (x, y, z) from the origin to the distance of p^* from the origin. The algorithm is:

1. find spherical coordinates (θ, ϕ) of coordinate (x, y, z)

2. search to find point $Q_d(u^*, v^*) = (x^*, y^*, z^*)$ with the same spherical coordinates as (x, y, z)

3. $(u, v, t) = \left(u^*, v^*, \frac{\sqrt{x^2+y^2+z^2}}{\sqrt{x^{*2}+y^{*2}+z^{*2}}}\right)$.

Note that t (proportional to depth) is necessary for occlusion testing and clipping.

Explanation is necessary to describe the search technique from step two that finds the (u^*, v^*) coordinates from spherical coordinates. The first step of this search requires precalculation on the CPU. That is, for each outline and profile, we calculate $n+1$ points,

equally spaced by arc-length, along the curve. The value $n+1$ should be chosen to balance the accuracy needed for the number of pixels across the image without requiring excessive memory; a value of half the image's larger dimension works well. This list of ordered points is stored after converting each point into spherical coordinates, recording θ for outlines and ϕ for profiles (see Figure 9.15). This data, along with the attachment points for each curve (also in spherical coordinates), are passed to the graphics card for a given Q_d and only need to be recalculated when Q_d is changed.

The GPU, once provided with the list of the ordered points and p's spherical coordinates (θ, ϕ), needs to calculate (u, v). This begins by calculating u. In the case of a single outline we find the array entries i and $i + 1$ that bound θ and u is calculated by interpolating between these array values. Care must be taken in the boundary case when θ is between array entries 0 and $n - 1$. With multiple outlines the calculation must include the additional step of interpolating between outlines, based on ϕ and the interpolation function $g(\phi)$ from Equation 9.2. The calculation of v is similar, making use of the profiles and ϕ. The only key difference is the treatment of the boundary case. As profile curves are not closed and do not entirely surround the center of projection, there will be points beyond the first and last array entries that should be culled from the image by assigning values outside the range of the surface (i.e., $v \notin [0..1]$).

Seams. Seams refer to where the projection surface has been split to be flattened into an image. Without special handling, triangles that intersect this seam become spread across the image. This is the result of one of a triangle's vertices being projected to one side of the image, the other vertices projected to the other side, and the linear fill algorithm interpolating between them as shown in Figure 9.16. With geometry shaders, this problem can be handled by testing for triangles that overlap the seam. When an overlapping triangle is found, a copy of the triangle is created; the original is placed

entirely on the left side of the image, the copy on the right. This ensures that interpolation does not occur across the entire image but instead off the left and right sides of the image.

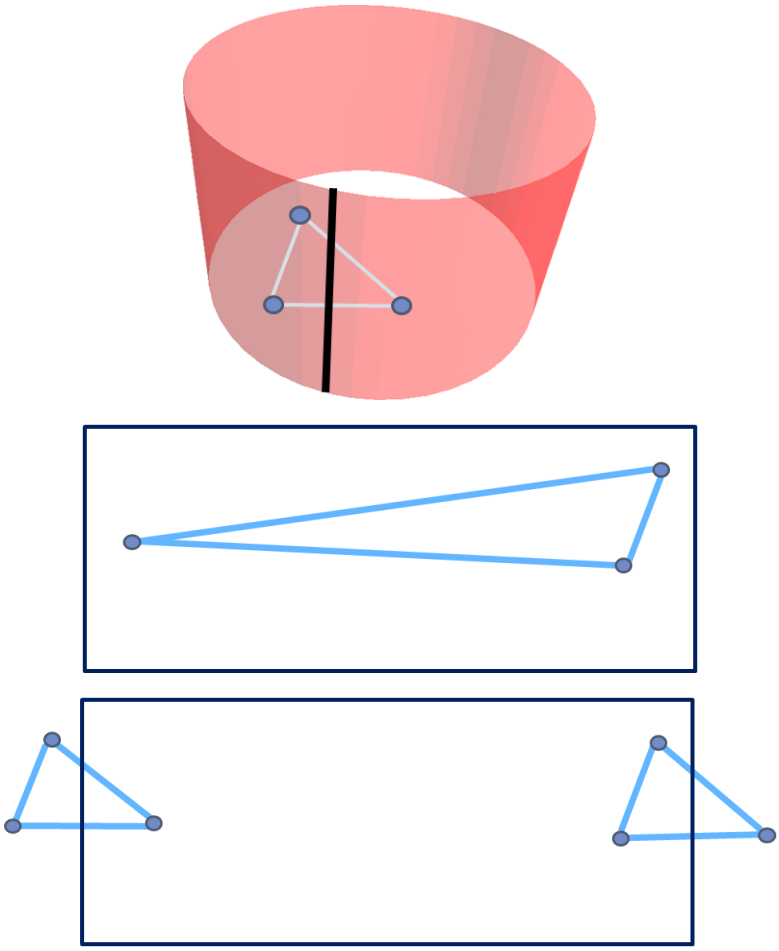


Figure 9.16: When a triangle is projected onto the projection surface such that it crosses the seam (top) the default filling algorithms will cause the triangle to be rendered across the majority of the image (middle). The problem is handled by creating two copies of the triangle, one of the left side of the image and the other on the right (bottom).

Accuracy. A problem with the accuracy of the image is caused by the standard filling algorithm's use of linear interpolation that does not properly handle the curving of triangle edges caused by nonlinear projection. The effects of this are similar to a coarse approximation of a curved surface and is especially noticeable when low polygon models are used. Gascuel et al. [GHFP08] provide an elegant solution for spherical projections where each triangle's maximum warped area, when projected is calculated

and then shaders are used to determine the extent of the curved triangle inside this area. While accurate, this technique relies heavily on the regular curving structure of spherical projections to calculate the warped area and thus would be difficult to adapt to our potentially irregular surfaces. The approach recommended is to use standard modeling software to apply planar subdivision uniformly to the scene. An alternative that may yield a more efficient result is to use an adaptive subdivision technique such as that of Pakdel and Samavati [PS07]. One could adaptively subdivide triangles with the geometry shader when an edge's midpoint deviates significantly from its linearly interpolated midpoint.

This rendering technique has real-time performance. In implementation, that includes handling of seams but not adaptive subdivision, frame rates of over 60 fps were achieved with a model of $\sim 100K$ triangles on an Intel Core2Duo 8400 with 4GB RAM and a NVIDIA 8800 GTS 640 MB graphics card.

9.4.3 Choice of Rendering Algorithm

As discussed at the start of this section, there are three choices in rendering algorithms: ray tracing, image resampling as described by Trapp and Döllner [TD08], and rasterization.

Before comparing these three techniques, a description of Trapp and Döllner's technique [TD08] is necessary. This technique renders single-center projections; that is, projections with a fixed view position. To render their algorithm first renders a cube-map from the view position. In their system user specifies a normal map, that in a ray casting sense, provides direction to the rays that sample the scene. In a second rendering pass, the normal map is used to resample the cube-map's pixels, creating the final projected image.

Choosing a rendering algorithm is a balance between speed and quality. For the

utmost in quality, ray tracing is the clear candidate. When real-time rendering is desired, the choice between Trapp and Döllner's cube map resampling [TD08] (Cube Maps) and Section 5.2's rasterization algorithm (Rasterization) depends upon the type of scene being projected, the shape of the projection surface, and on the desired resolution of the output image.

For scenes of extremely high numbers of polygon, it is possible that in rasterization the number of calculations performed per vertex on the GPU may outweigh the Cube Map's cost of a second image pass and image resampling. For low polygon scenes, the rasterization technique will either yield inaccuracies in image quality, as discussed in the preceding subsection or suffer slow downs due to planar subdivision of the scenery models. The image-based nature of the Cube Map technique will cause speed reductions when creating higher resolution output that will not be as noticeable in the object-space Rasterization process. Additionally high resolution output will cause Cube Map's reduction in final image quality, due to greater numbers aliasing artefacts associated with resampling the relatively lower resolution cube map.

The Cube Map technique will also suffer increasing inaccuracies as the shape of the projection surface diverges from that of the cube map, especially for areas of the surface corresponding to large arc-length where the cube map lacks sufficient resolution to provide necessary image detail for sampling. Lastly, the Cube Map technique will suffer when the arc-length parameterization of the cube is significantly different from the arc-length parameterization of the panoramic projection surface. Differences in this parameterization indicate under/over sampling scenarios that adversely affect image quality.

Figure 9.17: A cubic panorama modified by smoothing the corners of the square outline. Resulting projection surface (left) and image (right). Compare with Figure 9.8.

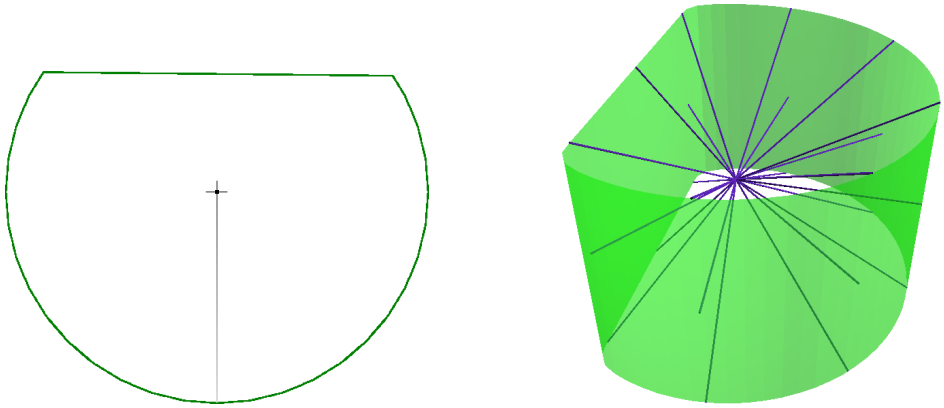


Figure 9.18: The outline (left) and resulting projection surface (right) used to create the image shown in Figure 9.2.

9.5 Results

This section provides several examples of how panoramas can be tailored to create unique images.

The first example presented in Figure 9.17 begins with the square outline that produces a cubic panorama. However this outline is altered by rounding off the corners of the square to avoid the sharp discontinuities normally present in cubic panoramas. The produced image provides views down each of the streets in perspective with a smooth transition between each of the four directions.

In motivating this work the possibility was mentioned of altering a cylindrical panorama to contain an area of perspective projection. Figure 9.2 and 9.18 provides an example

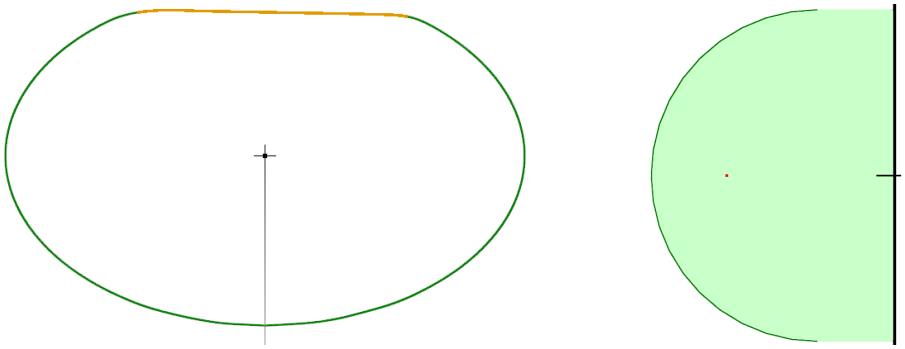


Figure 9.19: The outline (left) and profile (right) used to create the changes in Figure 9.20. The flattened area of the outline has been colored orange for emphasis.

of this where the outline has been flattened to make clear the rectangular nature of the building in the scene.

The next example begins with the goal of creating a panorama depicting a small hallway that includes the ceiling, floor, and walls. Figure 9.20 displays the starting point, a spherical panorama. From this point, one could imagine there is a user who wishes to change two aspects of this image: first this person would like to minimize the outward bulging of the wall in the middle of the image, secondly they want to reduce the amount of distortion along the top and bottom of the image that is caused by the poles of the sphere. The first problem can be addressed by changing the outline, flattening it slightly into an ellipse. In the resulting image this will horizontally compress the walls and expand the ends of the hallways. The second aspect is handled by translating the outline away from the axis so that the top of the surface will project a circle rather than a single point.

The last example, a panorama of a train station, is shown in Figure 9.21. In this example, the outline was made to be cylinder-like for the bottom third of the image in order to minimize the presence of the dark railway bed. The top of the outline is a hemi-circle translated away from the axis. This provides a spherical projection of the roof and

Figure 9.20: Altering a spherical panorama (top) of a hallway to produce customized panoramic projections (middle and bottom).

emphasizes its arching nature. The bottom half of the hemi-circle causes elements of the train station at pedestrian eye level to be vertically expanded in the image.

9.6 Applications

Aside from use of this framework to specify unique panoramas, there are several possible applications that take advantage of our panoramic projection surfaces.

Animated Panoramas. The first such application is in creating animated panoramas. Frames from such an animation are shown in Figure 9.23. An example of a situation where this might be useful is in providing a simulation of perceived peripheral vision for a moving viewpoint. In such a simulation, as the viewpoint moves faster, the center of projection within a circular outline is moved towards the front edge (i.e., opposite the seam) of the outline. This causes the world in front of the viewer to enlarge in the created image and areas beside and behind the viewer to shrink; simulating a change in focus when moving quickly.

Panorama Reprojection. Another possible application lies in reprojecting panoramas; using existing panoramic images to appear is if created by a different projection surface. That is, we can resample existing panoramic images to appear as if created by a different panorama. This application operates much like how Trapp and Döllner [TD08] resample cube maps, however we are not restricted to resampling cube-map images. The steps of this conversion are:

1. create a projection surface $Q_1(u, v)$ that approximately corresponds to the projection used to produce the image
2. use $Q_1(u, v)$ to find spherical coordinates for each pixel
3. use $Q_2(u, v)$ to determine the resulting spherical coordinates in each pixel of image produced by the new panorama
4. use both sets of spherical coordinates to sample and determine the pixel values.

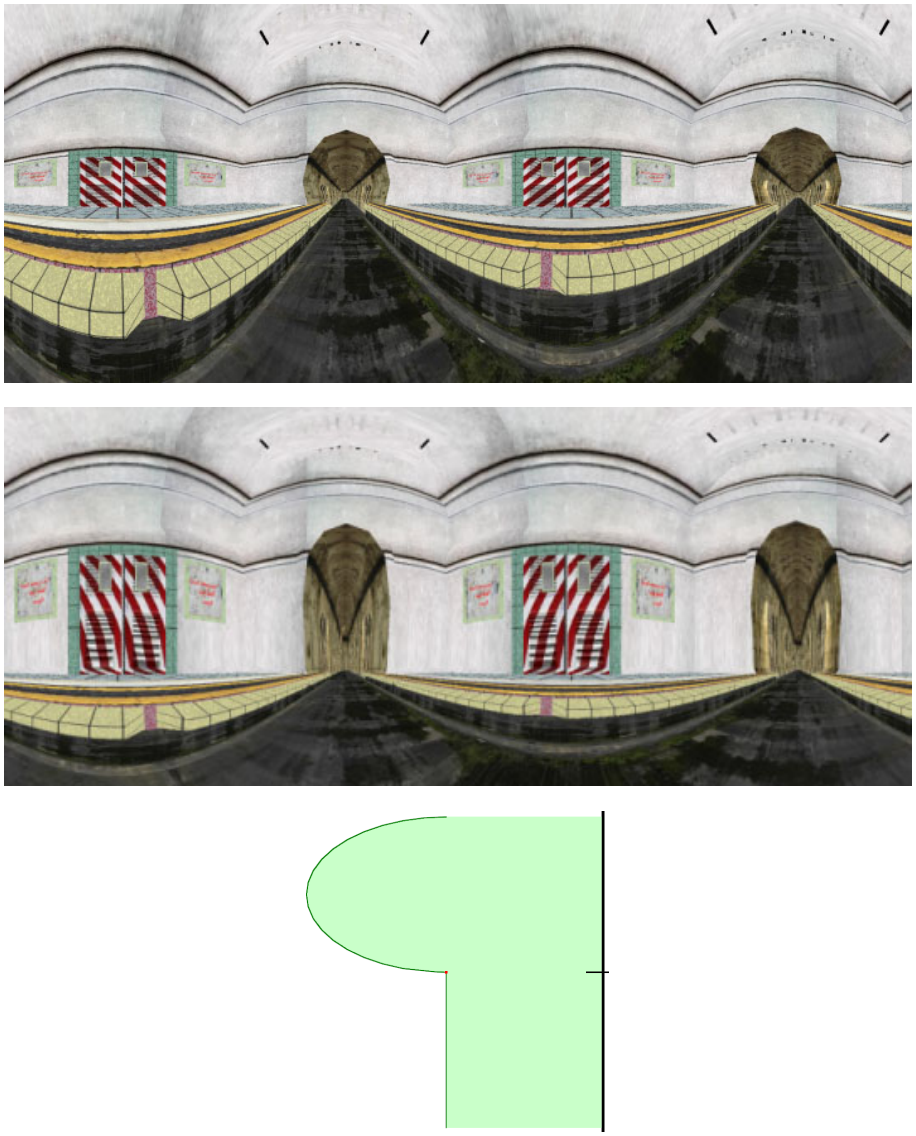


Figure 9.21: Altering a cylindrical panorama (top) to a customized panorama (center) that vertically stretches the scene at eye level as is evident in the scene's doors. The projection has also been designed to reduce the amount of image displaying the railway floor while retaining the arched roof. The outline in the altered panorama is a circle, the profile is shown in the bottom image.

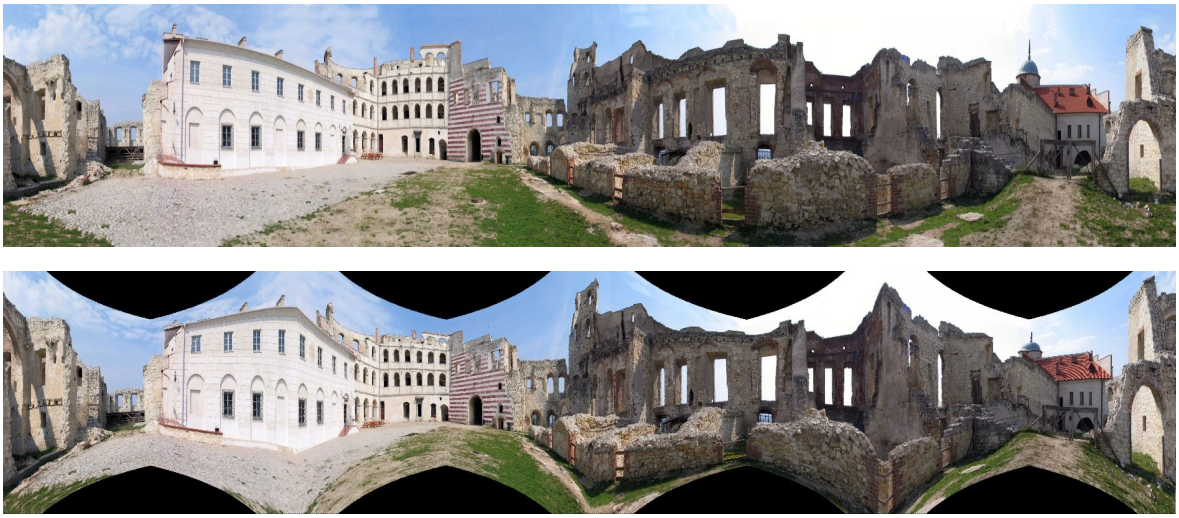


Figure 9.22: Left: a cylindrical panorama created from stitched together camera images. Right: reprojection to meet the specification of the panorama shown in Figure 9.12.

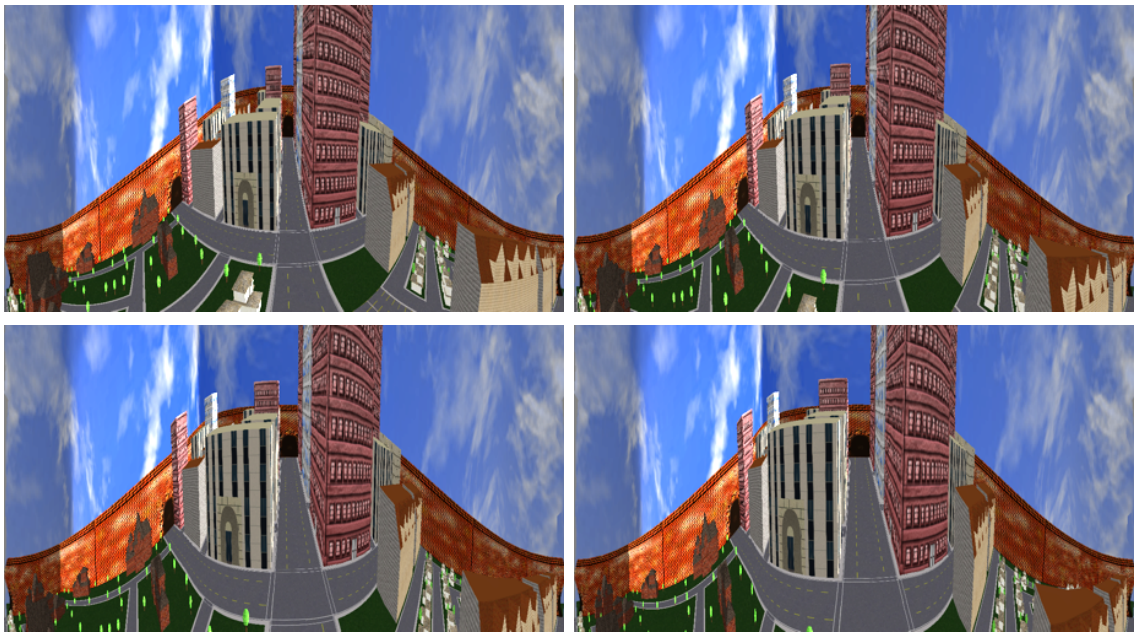


Figure 9.23: Four frames of an animation transitioning between a cylindrical panorama (top left) and the projection shown in Figures 9.2 and 9.18 (bottom right).

Figure 9.22 provides an example where a cylindrical panoramas is reprojected to another custom-made panorama. The areas sampled outside that of the original cylindrical panorama have been colored black.

Interactive Local Editing. The last application we will discuss is an interactive tool for local editing of panoramic projections. With the described realtime rendering technique it is possible to display and change the arc-length panorama surface interactively in image space.

In our proof of concept application the user begins with any arc-length panoramic surface and then drags the mouse over the image, specifying a rectangular area that they want altered to be projected in perspective. This change is accomplished by calculating the rectangular extent in spherical coordinates. These coordinates are used to extract outline and profile iso-curves from the projection surface at boundaries of the specified area. We extract four outlines and four profiles, one corresponding to each boundary of the rectangle and another slightly inside the rectangle to provide an area of interpolation between the original panorama and the perspectively projected area. The interior outlines and profiles are flattened to produce a flat area on the surface while the boundary curves maintain the original surface (see Figure 9.25).

These new outline and profile specifications are added to the surface definition, relayed to the geometry shaders, and the specified alteration is instantly displayed in realtime. Additionally this surface definition can be used to produce a high quality ray traced version of the panorama (Figure 9.24) This technique can easily be extended to allow other local alterations to be made to a panorama's image.

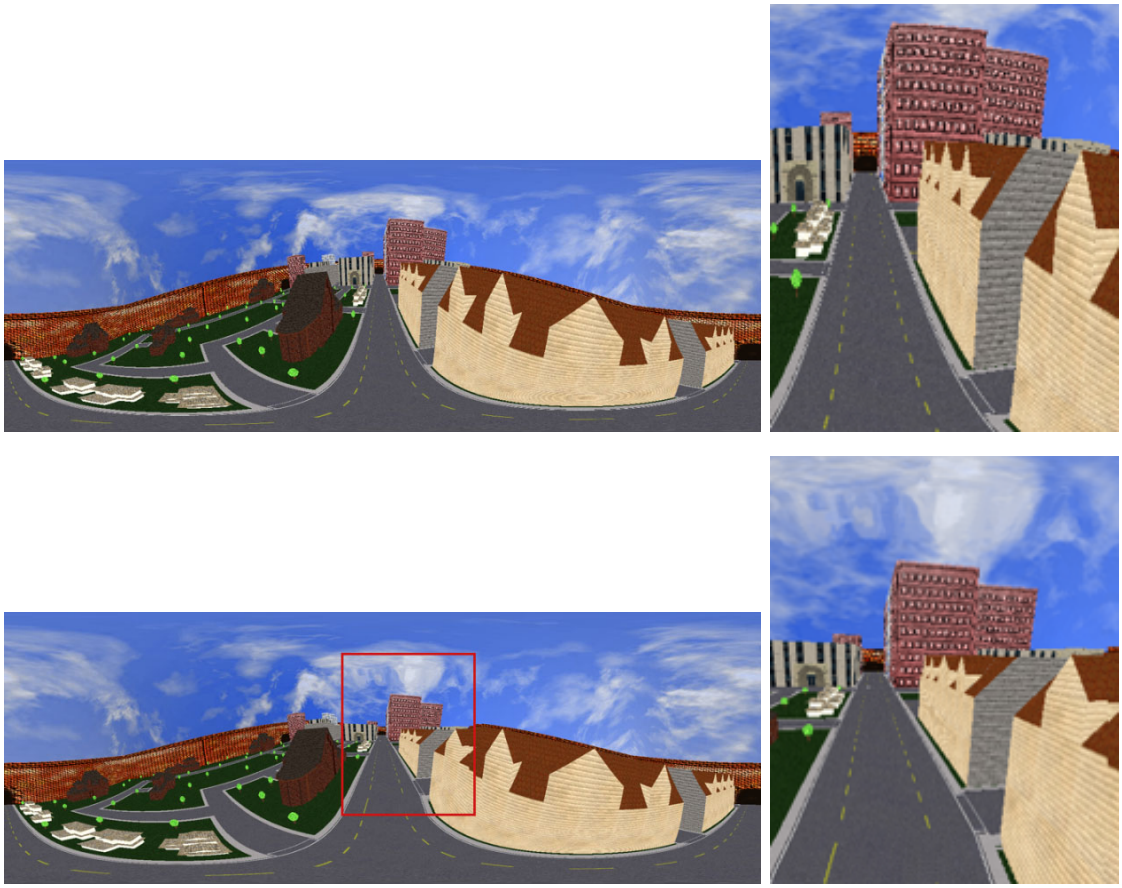


Figure 9.24: Interactive editing a panorama in image space. Top: the original spherical panorama. Bottom: the altered panorama. The red lines indicate the altered image region. Right: enlarged comparison of altered regions.

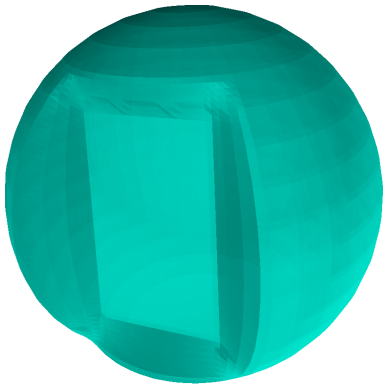


Figure 9.25: The arc-length parameterized surface produced by the interactive process shown in Figure 9.24.

9.7 Conclusion

This chapter has demonstrated how Flexible Projection can be adapted to create custom panoramas for a variety of effects. The described interface is helpful for Flexible Projection in general due to the explicit control over the surface's parameterization, the sampling of the scene, and the consequent image characteristics.

Additionally use of these projection surfaces parameterized by arc-length creates a relationship between the easily visualized surface and the projection behavior in the resulting image. This allows for creation of a wide variety of panoramas in a visual manner and draws upon existing modeling tools and conventions. This arc-length parameterization also seems to hold potential in adaption to other parametric surfaces for use with Flexible Projection.

This chapter also described specific techniques for adapting Flexible Projection's existing rendering approach to the unique challenges of rendering full-view panoramas in realtime.

Chapter 10

Conclusion

The majority of computer graphics is concerned with presenting three dimensional environments on two dimensional media. Understanding the process of projection is crucial to the success of the resulting images, both in terms of controlling how 3D space is expressed, as well as in communicating this process so that image creators can effectively apply this control. This thesis contributes to understanding projection by exploring the unification of a wide variety of projections into a single framework.

The Flexible Projection Framework encourages construction of projections in a geometric fashion, allowing existing modeling approaches and tools to be adapted to constructing projections. Additionally, the geometric nature of this framework allows the operation of the projection to be described visually, rather than just mathematically or through an input-output feedback loop.

The viability of Flexible Projection as a framework has been demonstrated through its use in:

- Reproducing a wide variety of projections from current literature,
- Interpolating between and extrapolating beyond existing projections,
- Developing animated cameras,
- Recreating an artistic projection and applying this projection to other scenes,
- Creating an art installation,
- Adapting it for the express purpose of creating customizable panoramas.

10.1 Contributions

This research has made several contributions.

10.1.1 A Projection Framework

The framework described by this research is able to represent diverse types of projections: linear, line preserving, nonlinear, projections with linear projectors, and projections with nonlinear projectors. We have shown how a wide variety of seemingly distinct projections from the literature relate to one another and can be accurately reproduced within our framework. This unified representation of diverse projections allows for easier changes or comparison between projections, as well as development of algorithms that work for a wide variety of different projections rather than only being applicable to a single type. Naturally, developing algorithms that can be applied to all possible projections is difficult and sometimes not possible. But even where it is not possible, such as in the scanline rendering algorithm described in Chapter 7, Flexible Projection encourages development for specific classes of projection rather than for specific point solutions such .

This framework also encourages projections to be described in a geometric fashion. This provides the ability to easily visualize the projections as well as to create projections using existing and well-understood modeling tools such as parametric surfaces and volumes.

10.1.2 Rendering Algorithms

Rendering nonlinear projection is challenging. This thesis has contributed two different approaches to meet this challenge.

Quadratic Ray Casting

While previous nonlinear ray tracing techniques created images by approximating nonlinear rays with ray segments, we have taken the more direct approach of analytically determining intersections for quadratic curves. Additionally, we have shown that the time costs of doing so are comparable, if not improved, over that of previous techniques.

Our approach of explicitly-defining nonlinear rays has not been seen before. Previous works in nonlinear ray tracing literature, while occasionally mentioning the possibility of explicitly defined curves [Gr5], have exclusively made use of implicit curves derived from vector fields or other dynamic systems.

Scanline Rendering

At this time, scanline rendering remains the primary means of achieving realtime rendering of 3D environments. Thus, in order to allow the interactive use of nonlinear projection some means of accomplishing scanline rendering is necessary; without interactive results and feedback nonlinear projection is frustrating to explore.

While our scanline rendering algorithms remain unable to execute all of the projections made possible in our framework, we have shown that such algorithms can be applied to projections with linear and nonlinear projectors. While other techniques have made use of scanline rendering for multi-camera projections [CS04, HWSG06] we have contributed several useful refinements: using programmable graphics hardware shaders to override the projection matrix with nonlinear projection equations; using geometry shaders to handle panorama seams; use of fast numerical approximations rather than projection equations (as shown for Panoramas in Chapter 9); and nonlinear depth-mapping for distance correction interpolation for inverse perspective (Appendix A).

10.1.3 Distinction Between Linear and Nonlinear Projections

In Chapter 2 we took great care in distinguishing the boundary between linear and nonlinear projections, rather than relying on the assumption that, if a projection is not perspective or parallel projection, it is nonlinear. This distinction is important in computer graphics as it differentiates between those projections that can be easily integrated into the standard pipeline and those that cannot. This distinction also assists in realizing which aspects of the standard rendering pipeline must be accommodated if one wishes to scanline render nonlinear projections.

10.1.4 Arc-Length Parameterization

A smaller contribution is the concept of arc-length parameterization. That is, parameterization of surfaces and possibly volumes that explicitly depend on the object's shape. We have shown how, in the context of defining projection volumes, arc-length removes the unseen formulas controlling parameterization and makes the resulting image entirely dependent on the shape of the viewing volume. This is useful in communicating the relation between the behavior of a projection and the shape of its viewing volume.

10.2 Future Work

There are several areas for future work related to this thesis: interfaces for Flexible Projection; advancement of nonlinear ray tracing; improvements of scanline-based rendering; and applications.

10.2.1 Interfaces

A major area for future exploration lies in exploring new interfaces and variations for creating Flexible Projection viewing volumes.

While the interfaces described in Chapter 5 concentrated on specific geometric elements of defining a viewing volume, Section 9.6 introduced the possibility of providing input directly in image space that, transparently to the user, causes modification of the viewing volume and, in-turn, modifies the image. This type of approach addresses a shortcoming of the current interface in that it relies heavily on direct manipulation of 3D model. For individuals outside without modeling knowledge, who simply want "this" part of the image to look like "that", or appear "over there" the current system requires a great deal of learning and indirect experimentation. Even for individuals that are familiar with these modeling tools, often many iterations of trial and error are required to accomplish image-based tasks. Development of an image-based interface that would algorithmically specify the necessary viewing volume to match the human-provided image space criteria could prove very beneficial. A more developed system could allow, for instance, translation of objects, specification of vanishing points or other construction lines, or changing viewing directions. While potentially powerful, any such image-based interface will require a strictly-defined scope of possible operations where the necessary modification to the viewing volume is well understood.

A drawback of the control surface interface (Section 5.1) is that while the first and last surfaces are interpolated by the Bézier curve projectors, the in-between surfaces that allow the curving of the projectors are only approximated. A movement to an alternative curve representation or a constraint based solution that would provide the approximate surface necessary to interpolate the specified in-between surface (similar to the modeling technique of Pusch and Samavati [PS10]) would resolve this difficulty.

The current projector-based interface (Section 5.2) only allows users to specify a 5×5 grid of projectors rather than being able to specify control over arbitrary projectors within the volume. A constraint based solution such as the one described in the previous paragraph also holds the potential solution to this problem. That is, user defined

projectors can be treated as constraints that an optimization algorithm uses to choose the control points of a trivariate B-Spline or Bézier volume that cause the user defined projectors to be closely approximated.

10.2.2 Nonlinear Projectors

While the description of this Framework does not exclude the possibility, we have not delved into the potential for the use of piecewise curve segments of higher order than linear. One type of projection that can only be created in a limited sense with the currently implemented system, is a projection that begins with a perspective projection close to the near surface and then smoothly blends to a different perspective projection towards the far surface (such as in Figure 8.7). Use of piecewise quadratic projectors would allow both perspective areas of the volume as well as the transition to be more precisely and easily controlled.

Allowing for piecewise quadratic projectors opens possibilities for time savings when ray-casting higher order curves. As was shown in Sub-Section 7.2.2, the intersection calculation cost of a quadratic curve seems to lie somewhere between that of three to six linear intersections. Better understanding of these time costs may lead to an algorithm where higher order curves are approximated by both linear and quadratic curve segments to provide a higher accuracy approximation for an equivalent or better calculation time. A related area of interest lies in determining the cost of an analytic intersection test for cubic curves to see whether it may prove efficient in comparison to approximation by piecewise quadratic curves.

Quadratic-based ray casting and tracing also provides other interesting issues. One such is the investigation into adaptation of existing linear ray optimization techniques for quadratic rays. Techniques such as oct-trees rely on extremely inexpensive tests that determine whether linear rays intersect particular levels of space subdivisions. Such tests

would be necessarily more expensive for quadratic curves and examination is necessary to determine whether these and similar techniques could successfully be adapted for this new scenario.

10.2.3 Scanline Rendering

While the scanline rendering algorithm outlined in Section 7.3 is suited for a variety of useful projections, it is not capable of rendering all possible varieties of Flexible Projections. An approach that bears potential for expanding the range of projections that can be rendered in this fashion is bypassing analytical solutions for Q^{-1} and instead making use of numerical approximations like the one demonstrated in Section 9.4. The main difficulty in such an approach lies in constraining the possible projections in such a way that will allow the numerical approximation to be performed efficiently while including the largest possible number of useful Flexible Projection volumes.

10.2.4 Applications

Beyond the applications previously described there are a wide variety of areas where Flexible Projection can be applied.

One possibility lies in making use of curved projectors in the creation of stereoscopic projections. Stereoscopic images are limited in the amount of depth that can be perceived. In his research into this problem Holliman [Hol04] has described a projection designed to preserve the depth perceived in an area of interest while compressing the depth outside this area. This projection is essentially the result of concatenating three perspective projections together to create a shallow compressed region, a focus region where accurate depth that preserves scale is perceived, and the deepest area of the volume where depth is again compressed. However the transition between these regions is piecewise, causing

sharp changes in perceived depth at the boundary. An ideal application of nonlinear projectors is their use in smoothing these transitions between regions.

To extend the idea of animated cameras, projections can also be made to react to elements other than time and key-framing. For instance one could imagine something similar to the described focus of attention examples described in Chapter 8 but coupled to gaze tracker that would expand the image-space areas of an image that the viewer looks at the longest.

Another interesting possibility is further investigating the possibilities of working within nonlinearly projected volumes. Can nonlinear projections assist in the exploration of complex data volumes such as those used in seismic and medical analysis? Are there times where working directly within nonlinear projections of these volumes might be useful? I.e., is there some sort of nonlinear projection that would make modeling easier through not having to rotate the viewpoint as much? It seems likely that a projection that preserves angles, such as an angular fisheye projection, might provide an interesting starting point for such possibilities.

Lastly, the projections demonstrated throughout this thesis have concentrated on producing flat, rectangular images designed for display on the page or monitor. However, with the ongoing development of LCD and e-ink displays we are seeing the beginnings of displays that provide images upon curved surfaces. Use of a Flexible Projection where the near surface corresponds to the shape and parameterization of the pixels across such a curved display may provide a better experience of 3D environments with these novel devices.

Bibliography

[AAC+06] A. Agarwala, M. Agrawala, M. Cohen, D. Salesin, and R. Szeliski, "Photographing long scenes with multi-viewpoint panoramas," in *SIGGRAPH '06*. ACM Press, 2006, pp. 853–861. 48, 94, 95, 154

[AG95] P. Acquisto and E. Gröller, "A distortion camera for ray tracing," Institute of Computer Graphics and Algorithms, Vienna, University of Technology, Tech. Rep. TR-186-2-95-05, Mar 1995. 34, 84

[AL07] A. Adams and M. Levoy, "General linear camears with finite aperture," in *Proc. Eugraphics Symposium on Rendering*, 2007. 36, 39

[AMHH08] T. Akenine-Mller, E. Haines, and N. Hoffman, *Real-Time Rendering 3rd Edition*. A. K. Peters, Ltd., 2008. 170

[AZM00] M. Agrawala, D. Zorin, and T. Munzner, "Artistic multiprojection rendering," in *Proceedings of the Eurographics Workshop on Rendering Techniques 2000*. Springer-Verlag, 2000, pp. 125–136. 6, 48, 49, 98, 100

[BCS+09] J. Brosz, S. Carpendale, F. Samavati, H. Wang, and A. Dunning, "Art and nonlinear projection," in *Bridges 2009: Mathematical Connections in Art, Music and Science*, 2009. 140

[BS10] J. Brosz and F. Samavati, "Shape defined panoramas," in *Proceedings of Shape Modeling International*. IEEE, 2010, pp. 57–67. 150

[BSCS07] J. Brosz, F. Samavati, S. Carpendale, and M. C. Sousa, "Single camera flexible projection," in *Proceedings of the 5th international symposium on non-photorealistic animation and rendering*. ACM, 2007, pp. 33–42. 56

[CAA09] R. Carroll, M. Agrawal, and A. Agarwala, "Optimizing content-preserving projections for wide-angle images," in *SIGGRAPH '09: ACM SIGGRAPH 2009 papers*. ACM, 2009, pp. 1–9. 154, 168

[CCF97] M. S. T. Carpendale, D. Cowperthwaite, and F. Fracchia, "Extending distortion viewing from 2d to 3d," *IEEE Computer Graphics and Applications*, vol. 17:4, pp. 42–51, Jul/Aug 1997. 42, 43, 91, 134

[Cen07] E. V. E. Center, "The panorama," http://www.edvec.ed.ac.uk/html/projects/panorama/, Accessed Oct 4, 2007. 150

[CF05] D. Claus and A. W. Fitzgibbon, "A rational function lens distortion model for general cameras," in *Proc. of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1. IEEE Computer Society, 2005, pp. 213–219. 36, 37, 38, 88

[CH03] J. P. Collomosse and P. M. Hall, "Cubist style rendering from photographs," *IEEE Transactions on Visualization and Computer Graphics*, vol. 09, no. 4, pp. 443–453, 2003. 47, 48, 98

[CM01] S. Carpendale and C. Montagnese, "A framework for unifying presentation space," in *UIST '01: Proceedings of the 14th annual ACM symposium on User interface software and technology.* ACM Press, 2001, pp. 61–70. 42, 43, 69, 91

[Com99] B. Comment, *The Panorama.* Reaktion Books, 1999. 150

[Coq90] S. Coquillart, "Extended free-form deformation: a culpturing tool for 3d geometric modeling." *SIGGRAPH Comput. Graph.*, vol. 24, no. 4, pp. 187–196, 1990. 70

[Cow00] D. J. Cowperthwaite, "Occlusion resolution operators for three-dimensional detail-in-context," Ph.D. dissertation, Simon Fraser University, August 2000. 42

[CP78] I. Carlbom and J. Paciorek, "Planar geometric projections and viewing transformations," *ACM Comput. Surv.*, vol. 10, no. 4, pp. 465–502, 1978. 12, 18, 19, 21, 22, 25

[CS04] P. Coleman and K. Singh, "Ryan: rendering your animation nonlinearly projected," in *NPAR '04: Proceedings of the 3rd international symposium on Non-photorealistic animation and rendering.* ACM Press, 2004, pp. 129–156. 50, 51, 98, 100, 103, 104, 121, 188

[CSSJ05] J. J. Cherlin, F. Samavati, M. C. Sousa, and J. A. Jorge, "Sketch-based modeling with few strokes," in *SCCG '05: Proceedings of the 21st spring conference on Computer graphics.* ACM, 2005, pp. 137–145. 163

[Dar07] G. Dargaud, "Fisheye photography," January 2007, http://www.gdargaud.net/Photo/Fisheye.html. 69

[DK09] P. Degener and R. Klein, "A variational approach for automatic generation of panoramic maps," *ACM Trans. Graph.*, vol. 28, no. 1, pp. 1–14, 2009. 54, 154

[Far01] G. Farin, *Curves and Surfaces for CAGD: A Practical Guide*, 5th ed. Morgan Kauffman, 2001. 162, 163

[Fur97] C. A. Furuti, "Modified azimuthal projections," 1997. [Online]. Available: http://www.progonos.com/furuti/MapProj/Dither/ProjMAz/projMAz.html 45

[GG99] G. Glaeser and E. Gröller, "Fast generation of curved perspectives for ultra-wide-angle lenses in vr applications," *The Visual Computer*, vol. 15, no. 7-8, pp. 365–376, November 1999. 153

[GH97] R. Gupta and R. I. Hartley, "Linear pushbroom cameras," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 9, pp. 963–975, 1997. 15, 28, 29, 36, 37, 38, 39, 85

[GHFP08] J.-D. Gascuel, N. Holzschuch, G. Fournier, and B. Peroche, "Fast non-linear projection using graphics hardware," in *ACM Symposium on Interactive 3D Graphics and Games*. ACM, Feb 2008. 174

[Gla89] A. Glassner, *An Introduction to Ray Tracing*. Morgan Kaufmann, 1989. 110

[Gla00] ——, "Cubism and cameras: Free-form optics for computer graphics," Microsoft, Tech. Rep. MSR-TR-2000-05, January 2000. 6, 34, 84

[Gla04a] ——, "Digital cubism." *IEEE Computer Graphics and Applications*, vol. 24, no. 3, pp. 82–90, May-Jun 2004. 34

[Gla04b] ——, "Digital cubism, part 2," *IEEE Computer Graphics and Applications*, vol. 24, no. 4, pp. 84–95, Jul-Aug 2004. 34

[Gla11] ——, "Cubism and cameras: Free-form optics for computer graphics," Accessed March, 2011. [Online]. Available: http://www.glassner.com/andrew/cg/research/cubism/cubism.htm 35

[Gol02] R. Goldman, "On the algebraic and geometric foundations of computer graphics," *ACM Trans. Graph.*, vol. 21, no. 1, pp. 52–86, 2002. 2

[Gr5] E. Gröller, "Nonlinear ray tracing: Visualizing strange worlds," *The Visual Computer*, vol. 11, no. 5, pp. 263–274, May 1995. 46, 93, 109, 112, 117, 188

[Gre86] N. Greene, "Environment mapping and other applications of world projections," *IEEE Computer Graphics and Applications*, vol. 6, no. 11, pp. 21–29, 1986. 153

[HCS+07] P. M. Hall, J. P. Collomosse, Y.-Z. Song, P. Shen, and C. Li, "Rtcams: A new perspective on nonphotorealistic rendering from photographs," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 5, pp. 966–979, 2007. 36, 37, 39, 40, 41, 50, 88, 89, 98

[Hee72] P. A. Heelan, "Towards a new analysis of the pictorial space of vincent van gogh," *The Art Bulletin*, vol. 54, no. 4, pp. 478–492, December 1972. 143, 145

[Hoc06] D. Hockney, *Secret Knowledge: Rediscovering the Lost Techniques of the Old Masters*, 2nd ed. Viking Studio, 2006. 3, 23

[Hol04] N. S. Holliman, "Mapping perceived depth to regions of interest in stereoscopic images," in *Proceedings of SPIE-IS&T Electronic Imaging*, vol. 5291, 2004. 192

[HWSG06] X. Hou, L.-Y. Wei, H.-Y. Shum, and B. Guo, "Real-time multi-perspective rendering on graphics hardware," in *SIGGRAPH '06: ACM SIGGRAPH 2006 Sketches*. ACM Press, 2006, p. 79. 38, 188

[Ina91] M. Inakage, "Non-linear perspective projections," in *Modeling in Computer Graphics (Proceedings of the IFIP WG 5.10)*, 1991. 22, 26, 32, 53, 84

[Lan04] C. Landreth, "Ryan," Film produced by Copper Heart Entertainment and the National Film Board of Canada, 2004. 50

[Len04] E. Lengyel, *Mathematics for 3D Game Programming and Computer Graphics*, 2nd ed. Charles River Media, Inc, 2004. 25, 115, 116, 201

[Lev98] J. Levene, "A framework for non-realistic projections," Master's thesis, Massachusetts Institute of Technology, 1998. 33, 34, 48, 84

[LTJD08] H. Lorenz, M. Trapp, M. Jobst, and J. Döllner, "Interactive multi-perspective views of virtual 3d landscape and city models," in *11th AGILE International Conference on GI Science*, ser. Lecture Notes in Geoinformation and Cartography. Springer, 2008, pp. 301–321. 53, 54

[MGT00] D. Martin, S. Garcia, and J. C. Torres, "Observer dependent deformations in illustration," in *NPAR '00: Proceedings of the first international symposium on Non-photorealistic animation and rendering*. ACM Press, 2000, pp. 75–82. 53

[MPS05] C. Mei, V. Popescu, and E. Sacks, "The occlusion camera," *Computer Graphics Forum*, vol. 24, no. 3, pp. 335–342, 2005. 36, 40, 89, 90

[NAS02] NASA, "The blue marble: Land surface and ocean color and sea ice," NASA's Visible Earth, 2002. [Online]. Available: http://visibleearth.nasa.gov/view_rec.php?id=2430 45

[Nol10] C. Nolan, "Inception," Film, 2010, http://www.imdb.com/title/tt1375666/. 126

[OED11] OED Online, "projection, n," Oxford University Press, March 2011, accessed April 2011. [Online]. Available: http://www.oed.com/view/Entry/152272?redirectedFrom=projection 12

[PA06] V. Popescu and D. Aliaga, "The depth discontinuity occlusion camera," in *SI3D '06: Proceedings of the 2006 symposium on Interactive 3D graphics and games*. ACM Press, March 2006, pp. 139–143. 36, 41, 90

[Pir70] M. H. Pirenne, *Optics Painting and Photography*. Cambridge University Press, 1970. 23

[PPC97] J. A. Polack, L. A. Piegl, and M. L. Carter, "Perception of images using cylindrical mapping," *The Visual Computer*, vol. 13, no. 4, pp. 155–167, June 1997. 153

[PRAV09] V. Popescu, P. Rosen, and N. Adamo-Villani, "The graph camera," in *SIGGRAPH Asia '09: ACM SIGGRAPH Asia 2009 papers*. ACM, 2009, pp. 1–8. 51, 98

[Pri] PrintWiki, "Normalized device coordinate system," http://printwiki.org/Normalized_Device_Coordinate_System, accessed June 2010. 13

[PS07] H.-R. Pakdel and F. F. Samavati, "Incremental subidivision for triangle meshes," *International Journal of Computational Science and Engineering*, vol. 3, no. 1, 2007. 175

[PS10] R. Pusch and F. Samavati, "Local constraint-based general surface deformation," in *Proceedings of the International Conference on Shape Modeling and Applications (SMI 2010)*. IEEE Computer Society, Jun 21-23 2010, pp. 256–260. 190

[Rad99] P. Rademacher, "View-dependent geometry," in *SIGGRAPH '99*. ACM Press, 1999. 53, 153

[RB98] P. Radmacher and G. Bishop, "Multiple-center-of-projection images," in *SIGGRAPH '98: Proceedings of the 25th annual conference on computer graphics and interactive techniques*. ACM Press, 1998, pp. 199–206. 36, 40, 88

[RGL04] A. Román, G. Garg, and M. Levoy, "Interactive design of multi-perspective images for visualizing urban landscapes," in *VIS '04: Proceedings of the conference on Visualization '04*. IEEE Computer Society, 2004, pp. 537–544. 154

[Sal06] D. Salomon, *Transformations and Projections in Computer Graphics*. Springer-Verlag, 2006. 14, 25, 45, 63, 92, 158

[SB04] K. Singh and R. Balakrishnan, "Visualizing 3d scenes using non-linear projections and data mining of previous camera movements," in *AFRIGRAPH '04: Proceedings of the 3rd international conference on Computer graphics, virtual reality, visualisation and interaction in Africa*. ACM Press, 2004, pp. 41–48. 50

[SGS08] N. Sudarsanam, C. Grimm, and K. Singh, "Non-linear perspective widgets for creating multiple-view images," in *NPAR '08: Proceedings of the 6th international symposium on Non-photorealistic animation and rendering*. ACM, 2008, pp. 69–77. 50

[Shr06] D. Shreiner, *OpenGL programming guide*. Addison-Wesley, 2006. 13

[Sin02] K. Singh, "A fresh perspective," in *Graphics Interface*, May 2002, pp. 17–24. 25, 49, 50, 98, 100, 102

[SL96] J. Stam and E. Languénou, "Ray tracing in non-constant media," in *Proceedings of the eurographics workshop on rendering techniques '96*, Eurographics. Springer-Verlag, 1996, pp. 225–ff. 46

[Sny93] J. P. Snyder, *Flattening the Earth*, 2nd ed. University of Chicago Press, 1993. 45, 157

[SP86] T. W. Sederberg and S. R. Parry, "Free-form deformation of solid geometric models," in *SIGGRAPH '86: Proceedings of the 13th annual conference on Computer graphics and interactive techniques*. ACM Press, 1986, pp. 151–160. 70

[SPC04] R. Schmidt, E. Penner, and S. Carpendale, "Reconfigurable dsiplays," in *ACM Conference on Ubiquitous Computing, Workshop: Ubiquitous Display Environments; UBICOMP*. ACM Press, 2004. 143

[SS97] R. Szeliski and H.-Y. Shum, "Creating full view panoramic image mosaics and environment maps," in *SIGGRAPH '97*. ACM Press, 1997, pp. 251–258. 100, 153

[TD08] M. Trapp and J. Döllner, "A generalization approach for 3d viewing deformations of single-center projections," in *International Conference on Computer Graphics Theory and Applications (GRAPP)*, 2008, pp. 162–170. 35, 84, 85, 154, 169, 175, 176, 180

[TD09] ——, "Generalization of single-center projections using projection tile screens," in *Computer Vision and Computer Graphics. Theory and Applications*, ser. Communications in Computer and Information Science, A. Ranchordas, H. J. Arajo, J. M. Pereira, and J. Braz, Eds. Springer Berlin Heidelberg, 2009, vol. 24, pp. 55–69. 36

[Wat00] A. Watt, *3D Computer Graphics*, 3rd ed. Addison-Wesley Publishing Company Inc., 2000. 15, 24, 25, 80

[Wei00] D. Weiskopf, "Four-dimensional non-linear ray tracing as a visualization tool for gravitational physics," *VIS*, vol. 00, p. 12, 2000. 46, 47, 109, 112

[Wei11a] E. W. Weisstein, "Affine transformation," MathWorld – A Wolfram Web Resource, 2011, accessed February 2011. [Online]. Available: http://mathworld.wolfram.com/AffineTransformation.html 17

[Wei11b] ——, "Map," MathWorld – A Wolfram Web Resource, 2011, accessed April 2011. [Online]. Available: http://mathworld.woflram.com/Map.html 13

[WFH+97] D. N. Wood, A. Finkelstein, J. F. Hughes, C. E. Thayer, and D. H. Salesin, "Multiperspective panoramas for cel animation," in *SIGGRAPH '97*. ACM Press, 1997, pp. 243–250. 40, 125, 154

[Wik10] Wikipedia, "Bullet time," Accessed December, 2010. [Online]. Available: http://en.wikipedia.org/wiki/Bullet_time 126

[WM90] G. Wyvill and C. McNaughton, "Optical models," in *Proceedings of the eighth international conference of the Computer Graphics Society on CG International '90: computer graphics around the world.* Springer-Verlag, 1990, pp. 83–93. 33, 84

[WSE04] D. Weiskopf, T. Schafhitzel, and T. Ertl, "Gpu-based nonlinear ray tracing," *Computer Graphics Forum*, vol. 23, no. 3, pp. 625–633, 2004. 46, 112

[WW99] A. Wachowski and L. Wachowski, "The matrix," Film, 1999, http://www.imdb.com/title/tt0133093/. 126

[WZMK05] L. Wang, Y. Zhao, K. Mueller, and A. Kaufman, "The magic volume lens: An interactive focus+context technique for volume rendering," *VIS*, vol. 00, p. 47, 2005. 44, 92

[YCB05] Y. Yang, J. X. Chen, and M. Beheshti, "Nonlinear perspective projections and magic lenses: 3d view deformation," *IEEE Computer Graphics and Applications*, vol. 25, no. 1, pp. 76–84, 2005. 42, 69, 91

[YM04a] J. Yu and L. McMillan, "A framework for multiperspective rendering," in *15th Eurographics Symposium on Rendering (EGSR04)*, 2004, pp. 61–68. 51, 52, 94, 96, 154

[YM04b] ——, "General linear cameras," in *Computer Vision - ECCV 2004*, vol. 2. Springer Berlin / Heidelberg, 2004, pp. 14–27. 25, 37, 38, 51, 85

[ZB95] D. Zorin and A. H. Barr, "Correction of geometric perceptual distortions in pictures," in *SIGGRAPH '95.* ACM Press, 1995, pp. 257–264. 22, 36, 154, 160

[ZFPW03] A. Zomet, D. Feldman, S. Peleg, and D. Weinshall, "Mosaicing new views: The crossed-slits projection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 6, pp. 741–754, 2003. 36, 37, 38, 85

[ZMP07] L. Zelnik-Manor and P. Perona, "Automating joiners," in *Proc. of the 5th int. symp. on non-photorealistic animation and rendering.* ACM, 2007, pp. 121–131. 47, 98, 100

[ZMPP05] L. Zelnik-Manor, G. Peters, and P. Perona, "Squaring the circle in panoramas," in *IEEE International Conference on Computer Vision (ICCV)*, vol. 2, 2005, pp. 1292–1299. 154

Appendix A

Perspective Depth Mapping for Correct Interpolation

The mathematics outlined in this Appendix is based on that described by Lengyel [Len04, pp. 117-120].

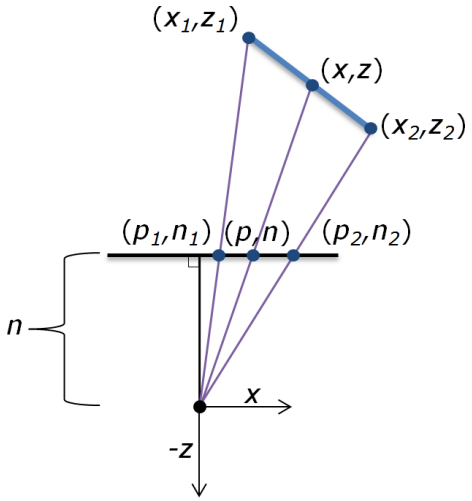


Figure A.1: Scanline rendering of a triangle. The blue line segment represents the relevant 2D cross-section of the triangle being rendered; the black line segment represents the image plane (the near surface of the viewing volume).

Consider that we are rendering a horizontal scanline of a triangle as shown in Figure A.1. In the two dimensions that we are concerned with (i.e., the x and z coordinates) the equation for this triangle is

$$ax + bz = c \tag{A.1}$$

where $c \neq 0$ since we assume that the triangle is not running through the origin (e.g., the center of projection). The point x, y on the triangle is projected to p, n on the image plane. Through similar triangles we can establish that

$$\frac{p}{x} = \frac{n}{z}. \tag{A.2}$$

By combining equations A.1 and A.2 we find

$$\begin{aligned} c &= a\left(\tfrac{pz}{n}\right) + bz \\ &= z\left(\tfrac{ap}{n} + b\right) \end{aligned} \tag{A.3}$$

and thus

$$\frac{1}{z} = \frac{ap}{cn} + \frac{b}{c}. \tag{A.4}$$

If the extremities of the line segments are labeled (x_1, z_1) and (x_2, z_2) then their respective projections on the image plane are (p_1, n_1) and (p_2, n_2) as shown in Figure A.1. Also let (p, n) be the coordinates of the point (x, z) when projected onto the image plane. If we assume that our scanline rendering algorithm is using linear interpolation to draw between (p_1, n_1) and (p_2, n_2) then

$$p = (1-t)p_1 + tp_2 \tag{A.5}$$

for some $t \in [0, 1]$. Substituting Equation A.4 into A.5 gives us

$$\begin{aligned} \tfrac{1}{z} &= \tfrac{ap}{cn} + \tfrac{b}{c} \\ &= \tfrac{ap_1}{cn}(1-t) + \tfrac{ap_2}{cn}t + \tfrac{b}{c} \\ &= \left(\tfrac{ap_1}{cn} + \tfrac{b}{c}\right)(1-t) + \left(\tfrac{ap_2}{cn} + \tfrac{b}{c}\right)t \end{aligned} \tag{A.6}$$

which, making use of Equation A.4 simplifies to

$$\frac{1}{z_1}(1-t) + \frac{1}{z_2}t. \tag{A.7}$$

This result indicates that the inverse of the z-coordinates is linearly interpolated across each scanline of any triangle when subjected to perspective projection.

Assume that we have values b_1 and b_2 that we wish to interpolate to determine the value b at our previous point x, z on the triangle's scanline. To perform this interpolation accurately the following relation

$$\frac{b - b_1}{b_2 - b_1} = \frac{z - z_1}{z_2 - z_1} \tag{A.8}$$

must hold true. From Equation A.7 we express z in terms of z_1 and z_2

$$z = \frac{1}{\frac{1}{z_1}(1-t) + \frac{1}{z_2}t}. \tag{A.9}$$

Combining Equations A.8 and A.9 and solving for b yields

$$b = \frac{b_1 z_2(1-t) + b_2 z_1 t}{z_2(1-t) + z_1 t}. \tag{A.10}$$

From this we can multiply the numerator and denominator by $\frac{1}{z_1 z_2}$ to isolate z

$$\begin{aligned} b &= \frac{\frac{b_1}{z_1}(1-t) + \frac{b_2}{z_2}t}{\frac{1}{z_1}(1-t) + \frac{1}{z_2}t} \\ &= z\left(\frac{b_1}{z_1}(1-t) + \frac{b_2}{z_2}t\right). \end{aligned} \tag{A.11}$$

With this equation b/z can be interpolated across the triangle. When rendering, graphics hardware interpolates $1/z$ for the current position in the scanline. The reciprocal is calculated through multiplying by b/z to determine the correct interpolated value of the vertex attribute b.

To save processing costs during rasterization we can map depth in the projection as a function of $1/z$ instead of mapping it linearly. This removes much of the rendering time cost of perspective correct interpolation since this mapping occurs per-vertex while scanline calculations introduce a cost per pixel. This mapping has the form

$$z^* = \frac{A}{z} + B. \tag{A.12}$$

We also know that this mapping should yield $n = 0$ and $f = 1$ where n and f are the perpendicular distances from the eye position to the near and far surfaces respectively. With these givens we can solve for A and B

$$0 = \frac{A}{n} + B, \quad 1 = \frac{A}{f} + B \tag{A.13}$$

that leads to the solutions

$$A = \frac{fn}{n-f}, \quad B = \frac{-fn}{n(n-f)}. \tag{A.14}$$

The resulting perspective depth mapping function is

$$z^* = \left(\frac{fn}{n-f}\right)\frac{1}{z} - \frac{fn}{n(n-f)}. \tag{A.15}$$

A.1 Inverse Perspective

As shown in Figure A.2 the situation for inverse perspective is quite similar to that of perspective.

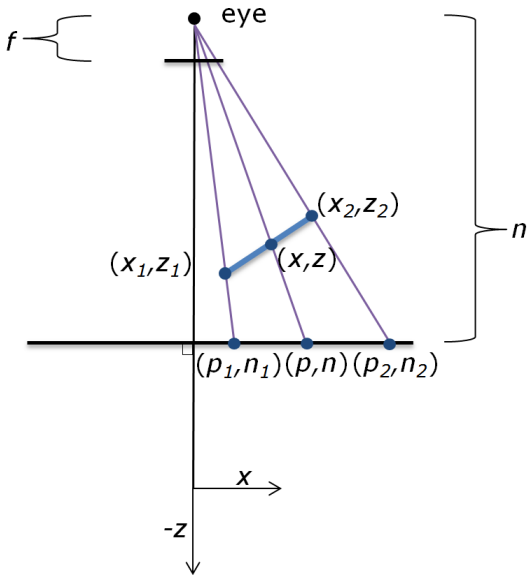


Figure A.2: Scanline rendering of a triangle after inverse perspective projection. The blue line segment represents the relevant 2D cross-section of the triangle being rendered; the black line segment represents the near plane and far planes.

As before the scanline of the triangle is $ax + bz = c$ and through similar triangles we can establish that

$$\frac{p}{x} = \frac{n}{e-z} \tag{A.16}$$

where e is the z coordinate of the eye position (behind the far plane) and n is the perpendicular distance from the eye position to the near plane. To simplify the notation

let $z' = e - z$. As with perspective by combining these equations we find

$$\begin{aligned} c &= a\left(\tfrac{pz'}{f}\right) + bz' \\ &= z'\left(\tfrac{ap}{n} + b\right) \end{aligned} \tag{A.17}$$

and thus

$$\frac{1}{z'} = \frac{ap}{cn} + \frac{b}{c}. \tag{A.18}$$

As with perspective when the scanline algorithm uses linear interpolation to draw between (p_1, n_1) and (p_2, n_2) it makes use of $p = (1-t)p_1 + tp_2$ for $t \in [0, 1]$. Substituting Equation A.18 into the linear interpolation gives us

$$\begin{aligned} \tfrac{1}{z'} &= \tfrac{ap}{cn} + \tfrac{b}{c} \\ &= \tfrac{ap_1}{cn}(1-t) + \tfrac{ap_2}{cn}t + \tfrac{b}{c} \\ &= \left(\tfrac{ap_1}{cn} + \tfrac{b}{c}\right)(1-t) + \left(\tfrac{ap_2}{cn} + \tfrac{b}{c}\right)t \\ &= \tfrac{1}{z'_1}(1-t) + \tfrac{1}{z'_2}t \end{aligned} \tag{A.19}$$

where z'_1 and z'_2 are $e - z_1$ and $e - z_2$ respectively.

As with perspective the result of equation A.19 indicates that the inverse of the z-coordinates is linearly interpolated across each scanline of any triangle when subjected to inverse perspective projection.

It is clear at this point that the calculation for an interpolated value b will be same as Equation A.11 with z', z'_1, and z'_2 substituted for z, z_1, and z_2.

The remaining task to is determine the depth mapping of inverse perspective as a function of $1/z'$ instead of mapping it linearly. As before, this mapping has the form

$$z^* = \frac{A}{z} + B. \tag{A.20}$$

We also know that this mapping should yield $z^* = 0$ at the near surface and $z^* = 1$ at the far surface. Thus we know

$$0 = \frac{A}{e-n} + B = \frac{A}{n'} + B, \quad 1 = \frac{A}{e-f} + B = \frac{A}{f'} + B \tag{A.21}$$

where n and f are the perpendicular distances between the eye and the near and far surfaces respectively. This leads to the solution

$$A = \frac{f'n'}{n' - f'} = \frac{(e-f)(e-n)}{f-n}, \quad B = \frac{-f'}{n' - f'} = \frac{f-e}{f-n}. \tag{A.22}$$

The resulting perspective depth mapping function is

$$z^* = \left(\frac{(e-f)(e-n)}{f-n}\right)\frac{1}{z} + \frac{f-e}{f-n}. \tag{A.23}$$

Appendix B

Pseudo Code for Quadratic-Triangle Intersection Test

The quadratic-triangle intersection test involves three steps: (1) rotating the quadratic by matrix R where r is the rotation necessary to to make the triangle parallel to $X = 0$; (2) converting the Bézier curve into the form $ax^2 + bx = c = 0$ and adjusting c by the triangle's x-intercept; and (3) solving the quadratic equation.

The pseudocode for this test is:

```
Function intersect
   Parameters
     P0, P1, P2 : Point;     // control points of Bezier quad
     R          : 3X3 Matrix // Rotation matrix of plane
     xint       : float;     // x-intercept of plane after rotation
   Return values
     hit        : Boolean;   // indicates intersection occurred
     t          : float;     // parameter value where intersection occurs


   begin
     // find rotated version of control points' x-coordinates
     // note P0[0] is the x-coordinate of control point 0.
     float x0 = R[0][0] * P0[0] + R[0][1] * P0[1] + R[0][2] * P0[2];
     float x1 = R[0][0] * P1[0] + R[0][1] * P1[1] + R[0][2] * P1[2];
     float x2 = R[0][0] * P2[0] + R[0][1] * P2[1] + R[0][2] * P2[2];


     // find a, b, c for quadratic equation
```

```
      float a = x0 - 2*x1 + x2;
      float b = 2 * (x1 - x0);
      float c = x0 - r.xintercept;

      // solve quadratic equation
      if (a != 0)
         float b2minus4ac = b*b-4*a*c;
         if (b2minus4ac >= 0)  // ensure real intercepts
            t1 = (-b + sqrt(b2minus4ac)) / (2*a);
            t2 = (-b - sqrt(b2minus4ac)) / (2*a);
            -> check that t1 & t2 are within [0,1]
             -> if both are, return hit=true, t=min(t1,t2)
             -> if one is not, return hit=true and the in range value
      else
         // linear case
         if (b != 0)
            t1 = -c/b;
            if (t1 >= 0 && t1 <= 1)
               return hit=true, t=t1;
      return hit=false, t=0;
   end
```

The above pseudo code can be made more efficient by storing and reusing values such as $sqrt(b2minus4ac)$ rather than recalculating them.