Interactive Control of Nonlinear Projection for Complex Animated Scenes

by

Patrick Coleman

A thesis submitted in conformity with the requirements
for the degree of Master of Science
Graduate Department of Computer Science
University of Toronto

Abstract

Interactive Control of Nonlinear Projection for Complex Animated Scenes

Patrick Coleman

Master of Science

Graduate Department of Computer Science

University of Toronto

2004

Linear perspective has long been at the foundation of most graphics systems. While it has proven to be effective for many applications, it offers limited creative control over the projection of a scene. *Nonlinear* projections allow for greater flexibility, but existing approaches do not adapt well to the interactive manipulation necessary for creative exploration. This thesis presents techniques for creating interactive nonlinear projections of complex scene geometry. A general nonlinear projection pipeline is presented that incorporates detailed control of projection, composition, illumination, and shadows. A high-level approach to creating nonlinear projections is then developed that builds upon this pipeline. This approach allows for direct specification of a nonlinear projection relative to an initial linear perspective view via the manipulation of feature proxies in the image plane. Additionally, procedural models of nonlinear projection are explored, including a hierarchal model and models that can respond to information such as object annotations, camera motion, and scene motion.

Acknowledgements

karan singh. for creating great ideas and letting us play with them. i couldn't say enough here. i've learned more from our in-depth discussions of both conceptual ideas and technical details than from any book, paper, course, or conference.

aaron hertzmann. again, his ideas are amazing and inspiring. as second reader, he greatly improved the writing in this thesis, and his suggestions have indirectly improved other pieces of writing along the way.

chris landreth. for his ability to create ideas in film that inspire and amaze. he also taught me how to write software that animators can actually use.

ravin balakrishnan, kyros kutulakos, demetri terzopoulos, and eugene fiume. for keeping dgp the challenging, heptic, and fun place that it is.

dave baas. for showing computer scientists how to animate a walk cycle.

leon barrett and cindy grimm. for collaborating on the material in chapter 3. it's always refreshing to bring outsiders in.

wael aboelsaadat, anastasia bezerianos, john hancock, and alan rosenthal. sysadmins are some of the most important people for allowing us all to get anything done, a fact we can so easily forget. thanks for all the patience and help along the way.

joe laszlo, mike mcguffin, mike wu. for their many editorial comments and help with video editing.

bruce weide. for the trip to OSU to talk about this stuff. and for that great first programming class so long ago.

all the demo victims, and the visitors from waterloo, york, montreal, and queens. for the feedback and interest.

mike neff. for the coffee stain near his desk, which is somewhere in this thesis.

chris trendall. for perpetually lending his optics book.

plantner and sostakowski, as always. you know exactly why.

brad winemiller, mike bernstein. i think you know why.

the self-described dgp vampires: mike wu (again), alex kolliopoulos, bowen hui, brad reid, and abishek ranjan. for making sure something was always happening in the lab, even at 4 am.

mike pratscher, marge coahran, mike tsang, maciej kalisiak, george elkoura, faisal qureshi, nigel morris, gonzalo ramos, xia liu, sam hasinoff, paul yang, kevin forbes, azeem lakdawalla, krista strickland, and everyone else in dgp who keep things going.

tom, pete, and chae. for carrying things around.

chris, paul, eric, juwanna, angelique... roommates/coworkers/friends later in the game.

martin, apurva, and bill. for letting me work on another challenging and fun problem.

alias. for donating the animation software that made life a lot easier.

pixar. again, for donating software that saved a lot of time. and an incredible place to intern. pun fully intended.

national film board of canada.

1389008 ontario, inc.

and, of course, s. b. he always his his fiftieth of a dollar. take off the mushroomhead t-shirt already.

Contents

Chapter 1

Introduction

Artists working with traditional media rarely use strict linear perspective. The ability to control and vary perspective in an image or animation allows artists to explore a much broader range of expression. Projection models that go beyond linear perspective can be a useful tool for affecting a viewer's perception of a scene. Digital artists, however, are usually constrained to work with the pinhole camera model common in computer graphics systems and image warping tools that work well for post-processing. The pinhole camera model has become popular as it not only approximates the appearance of the world as imaged by a camera or the human eye, but it can also be efficiently implemented as a linear transformation in graphics rendering systems. It is mathematically well-understood, and most interactive graphics systems are been built over a hardware implementation of the perspective projection pipeline.

Artists working with computer graphics cameras manipulate them similarly to how conventional cameras are operated. Parameters such as focal length and aspect ratio are manipulated as the focal length of a camera lens would be controlled to produce a desired field of view. On the other hand, the use of linear perspective in art is built upon techniques for laying out a few features that define the projection, such as vanishing points and horizon lines. Geometric constructions are commonly used to develop the

appearance of an object, given its desired position in the scene. This dependence of appearance on location is often broken to emphasize features or to depict objects from more appropriate or interesting vantage points.

1.1 Nonlinear Projections

The ability to incorporate the nonlinear projection of 3D scenes into a graphics system offers the potential for a great deal of expressive and creative exploration. Distorted projections can be used to influence a viewer's understanding of a scene by conveying emotion or developing a desired mood. Important content can be emphasized, and more information about complex shape can be conveyed than is possible with a single perspective view. The understanding of motion can also be affected, as perspective manipulations can incorporate more information about complex movement than a single view. Feelings of lightness or heaviness in motion can be influenced by projection, and spatial relationships among objects can be exaggerated.

The history of art, illustration, and film has many examples of how deviations from linear perspective have been used as tools for creative expression and emphasis. Historically, such deviations were used primarily as subtle manipulations (Chapter 2 presents a brief historical review), but non-realistic projections flourished in the art world during the 20th century. Artists such as Cezanne and Picasso intentionally broke traditional rules of perspective to present ideas in new ways, and the new medium of cinema developed with an strong emphasis an manipulating reality to affect the audience.

Deviations in perspective in both traditional media and film tend to take one of two forms. The first approach is to compose an image or scene with a small set of vantage points, showing each object or region of a scene from a point of view that best depicts its important visual qualities. An excellent example of this approach is Cezanne's *Still Life with Fruit Basket* (Figure 1.1). Note that while the table and many objects on it,

such as the closed jar to the left, are shown from a slightly elevated perspective, the larger, open jar is seen from a higher vantage point, revealing its interior. In contrast, the fruit basket on the right is depicted with an almost frontal point of view. Computer graphics researchers have recognized the creative potential of separating projection from composition; Agrawala et al. [3] imitated a variety of paintings, including Cezanne's *Fruit Basket* using multiple linear perspective renderings composed into a single image.

Figure 1.1: *Still Life with Fruit Basket.* Paul Cezanne. 1890. Each object is depicted from a vantage point that best depicts important visual content, as chosen by the artist.

The other common approach to deviating from traditional linear perspective involves continuously varying a point of view across an image. Picasso is perhaps best known for using this approach. His *Femme nue accroupie*(Figure 1.2) incorporates a continuously varying change in point of view. The left of the painting depicts the figure from a frontal viewpoint, and the right from a side viewpoint. The point of change in perspective is quite noticeable through the face, but the remainder of the body is depicted with a much smoother transition in point of view.

Manipulations of perspective that take advantage of the general nature of computer graphics cameras have had little exploration, however. Most approaches are based on

Figure 1.2: *Femme nue accroupie*. Pablo Picasso. 1959. Copyright 1959 the Estate of Pablo Picasso. This painting illustrates Picasso's use of continuously varying perspective within an image.

image manipulation with warping software or use 3D cameras that model live-action cameras. Nonlinear ray tracing has seen the broadest use, but has remained a niche area for technical experimentation with still images, primarily due to its computational expense. The nonlinear projection systems described in this thesis aim to make easier the creative exploration of nonlinear projections that are unique to the computer graphics medium. Central to their design is the incorporation of existing animation and rendering pipelines to allow for the interactive control and creative freedom they provide. In particular, the systems were motivated by the preproduction artwork and storyboard reel of the animated film *Ryan* [41]. This artwork illustrates a variety of projection techniques, including the use of continuously varying perspective and multiple projections within a single frame. Furthermore, the systems were designed to allow animators to bring to bear their skills and experience with working with a traditional computer graphics camera. By building projection up from a centralized *master* camera, animators can use and extend

beyond the communication techniques commonly used by cinematographers [37].

1.1.1 Nonlinear Projection in *Ryan*

This thesis presents a nonlinear projection system motivated by and designed for the production of the animated film *Ryan*. The production required the development of tools to support a wide variety of perspective manipulation effects. The pre-production artwork in Figure 1.3 illustrates a variety of projection approaches being incorporated into a single view. The film required that animators be able to work from a conventionally-animated, linear perspective camera, from which they could introduce distortions to perspective to achieve the appearance of a continuously varying viewpoint. Animators had to have the ability to independently specify the projections of different objects in a 3D scene, and frame composition had to remain independent of projection. Finally, the animators required a system that allowed them to interactively work with complex scenes.

The qualitative design of the system was particularly inspired by various pieces of preproduction artwork. Figure 1.3, in particular, illustrates in paint the types of effects that were required. First, the type of projection apparent in the image varies throughout the scene, incorporating both perspective and parallel linear projections, as well as a nonlinear projection of various straight edges. Second, these projections are disjoint, especially in the case of the column to the left of the image which appears as though seen from a much lower vantage point than the remainder of the scene. Despite these manipulations of perspective, two visual qualities of linear perspective remain to facilitate viewer understanding of the space. Any object, at a sufficiently small scale, has a visual appearance characteristic of linear perspective. In addition, the scene maintains a global compositional coherence that approximates true spatial relationships. As these two qualities facilitate viewer understanding, they were considered carefully when designing the system.

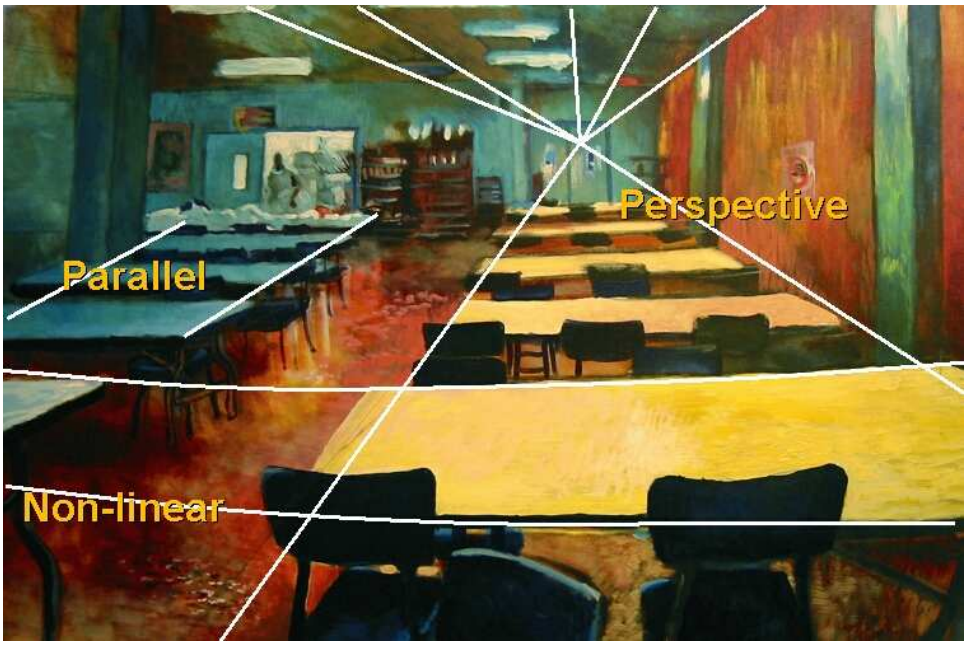


Figure 1.3: Preproduction artwork for *Ryan* incorporating an artistic combination of projection techniques. Qualities of linear perspective are maintained in the convergence of various parallel lines to a common vanishing point. The tables to the left of the image use an orthographic-like projection that maintains parallel line relationships. The horizontal edges of the tables closer to the viewer have a curved appearance characteristic of a nonlinear projection.

1.2 Limits of Current Approaches

While a variety of techniques have been presented in the graphics literature for creating nonlinear perspective images, they remain difficult to use, and few offer the creative power to interactively explore various projections. Existing approaches require off-line rendering systems, are difficult to control, or do not scale well with complex scenes. These limitations, and the complexities involved with specifying nonlinear projections, have precluded the use of general nonlinear projections in computer animation. Commercial and custom in-house animation systems are designed to leverage the power of graphics hardware, and have therefore been built upon the linear perspective pipeline. While

nonlinear projection effects have been developed for use in commercial rendering systems, they have seen little use, likely due to the lack of interactive tools that allow artists to explore various projections.

1.3 Contributions

To address the challenges described above, this thesis presents a model of projection that uses a multiple-camera control interface to allow for effective and intuitive exploration of nonlinear scene projections. The viewing and projection transformations of these cameras, in conjunction with per-camera spatial weight functions, define the nonlinear projection. The system uses deformation of scene geometry to allow for the incorporation of the model into existing graphics systems. Problems associated with rendering nonlinear projections are addressed, and two models of illumination are presented that incorporate the multiple view directions of the projection model. A constraint formulation is also introduced that allows a user to control locally and in detail the position, size, orientation, and visibility ordering of scene elements.

Building upon this system, this thesis then presents an approach for higher-level specification of nonlinear scene projections. This technique is analogous to the approach taken by artists working with physical media, whereby the desired projection of the 3D scene is specified on a 2D canvas. Using an exploratory view's projection as a starting point, the user can manipulate feature primitives such as points, bounding boxes, and sketched lines to indicate the desired projection. These manipulations are considered as composition constraints, and a nonlinear optimization solver is used to create a set of cameras that satisfies the constraints at the feature primitives. Implicit functions defined relative to these feature primitives are then edited to control the weight functions of the generated cameras to define the projection over the remaining space in the scene.

Additionally, procedural models of nonlinear projection are explored. A general-

purpose hierarchical framework is described, wherein cameras are introduced relative to a top level camera to introduce local manipulations of perspective. An initial exploratory model of hierarchical projection that uses this framework is then described. Central to the design of this hierarchical model, as well as the additional procedural models of projection that are presented, is the encapsulation of the model within a small set of intuitive parameters. Furthermore, all procedural models of projection are designed around a central master camera, allowing animators to use and build upon their existing cinematic skills. This approach allows for easy specification and animation of projections with predictable results, while automating the detailed control of cameras, camera weights, and constraints. A number of procedural control techniques are demonstrated, including projections sensitive to camera and object motion.

Chapter 2

Existing Approaches to Projection

Projection has always been used as a mathematical tool in the creation of computer generated images of 3D scenes. An integral part of the graphics pipeline, it defines how three-dimensional geometry is represented in a rendered image. Borrowing from techniques long used by artists to create naturalistic images, linear perspective has become the predominant computer graphics projection technique, as it approximates the function of both the human visual system and physical cameras. However, artists have long manipulated projection to better express the content of real scenes. Graphics systems have been developed that incorporate extended projection models not only for creative purposes, but also to aid in information visualization, to support systems for image-based rendering, and to provide an alternative data source for problems in computer vision.

This chapter presents a brief review of how projection has been used as an expressive tool by artists and computer graphics practitioners, and how extended projection models have been used to support vision and image-based rendering systems. A brief history of the development of perspective in art and science is first presented. The model of perspective projection common in graphics systems is then briefly described. This is followed by a survey of techniques that manipulate perspective for creative expression. These include post-rendering techniques such as image warping, as well as techniques

integrated into the rendering process such as stylized ray tracing, continuous multiperspective mosaics, and multiperspective projection. A brief description of how projection has been manipulated to facilitate interaction with information spaces is then discussed. The chapter concludes with a review of deformation techniques and an overview of camera control systems, which are relevant to both the nonlinear projection model described in Chapter 3 and the high-level projection control techniques presented in Chapters 4 and 5.

2.1 Projection in the Arts

(a) (b)

Figure 2.1: Historical Development of Projection. a) *The Announcement of the Angel to St. Anne.* Giotto Bondone. b) *The Baptistery of San Giovanni,* Filippo Brunelleschi, 1420.

The earliest use of perspective in images has been dated to about 4000 B.C., with the concurrent description of pinhole camera effects by Aristotle in Europe and use of camera obscura devices for sketching by Mohist philosophers in China [33]. The empirical use of perspective foreshortening is apparent in Greek and Roman art, but Western artists did not develop a clear understanding of linear perspective until the 15th century. Clear

errors in perspective are apparent in Western art created into the 1400's; a later example of this is Giotto's *The Announcement of The Angel to St. Anne* (Figure 2.1a). Filippo Brunelleschi is credited with the discovery of the mathematical rules for perspective projection, as he demonstrated in *The Baptistery of San Giovanni* (Figure 2.1b). These rules, along with a geometric construction for perspective foreshortening, were related by Leon Battista Alberti in his *Della Pittura* (*On Painting*), which led to the prolific use of linear perspective by Renaissance artists. The focus on naturalistic imagery in western art continued until the advent of the camera, after which artists such as Paul Cezanne rejected linear perspective in favor of more effective stylized depictions. Cubists such as Pablo Picasso explored the use of multiple points of view within a single image to incorporate views changing in space over time. M. C. Escher is also well-known for exploring multiperspective techniques, predominantly with a continuously changing view within an image [19]. Non-Western art has developed with a greater emphasis on stylization, and historically, it has incorporated a wider variety of perspective variation. Asian landscape paintings, in particular, are a well-known example of scenes that incorporate multiple views.

Western artists have long used optical devices such as camera obscura, mirrors, and camera lucida as projection tools to help capture not only correct perspective, but also accurate lighting and patterns on complex objects. Hockney argues that such techniques have been used to create locally accurate detail in larger images, which has a side effect of incorporating a variety of frontal views in a single image [33]. A historical example of this effect can be seen in Jan van Eyck's *Adoration of the Lamb* (Figure 2.2). Hockney has explored this technique in his own photomosaics, most notably *Pearblossom Highway* and *Place Furstenberg, Paris.* The compositions are made up of many frontal viewpoints, representing the way humans understand objects, while also reducing overall perspective distortion.

Among the wide variety of styles that have developed, perspective is primarily used as

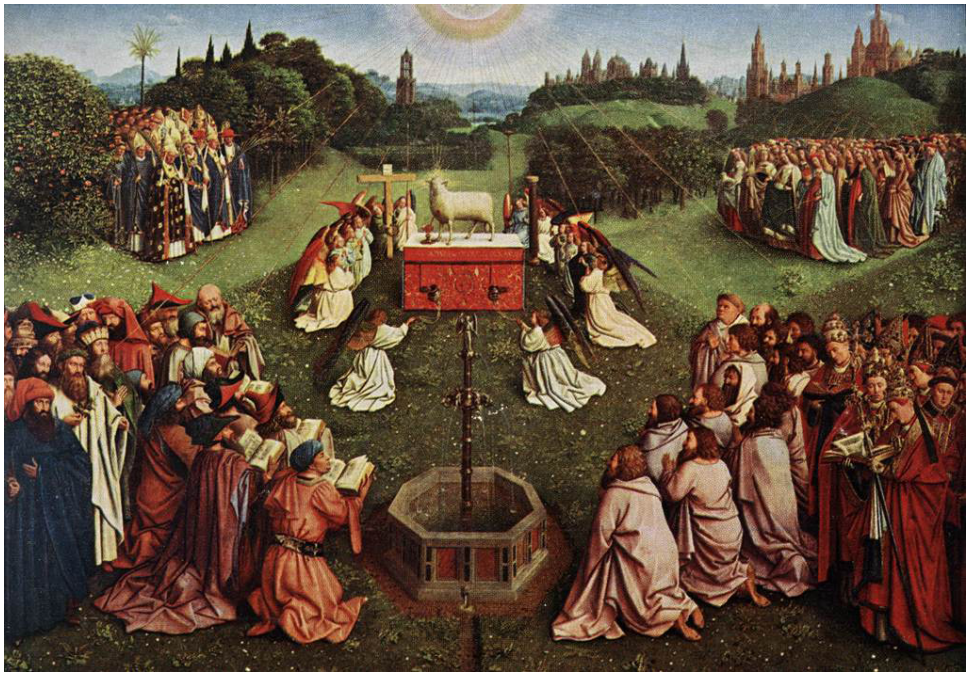


Figure 2.2: *Adoration of the Lamb* Jan van Eyck, 1425-1429.

a tool to present spatial relationships. Linear perspective, in particular, emphasizes the realistic depiction of geometric relationships to illustrate the shape of an object and its environment. Deviations from linear perspective commonly incorporate local qualities of linear perspective, retaining the shape and spatial understanding they allow. These manipulations of perspective most often incorporate either multiple linear perspectives on a per-object basis or large-scale perspective deformations that incorporate continuously changing views. Examples of the former include Giorgio de Chirico's *The Mystery and Melancholy of a Street* and Cezanne's *Still Life with Fruit Basket*, discussed in Chapter 1. Of particular importance in this approach is the independence of composition and projection. The latter approach can be seen more often in 20th century artwork, especially that of M. C. Escher and Pablo Picasso.

In the cinema world, both discrete and continuous variations of perspective have seen quickly growing popularity. False perspective set constructions are commonly used to make objects in a set appear from alternate points of view to compensate for available

space. Sets have also been built to present the appearance of distorted perspective, for example, in Robert Wiene's 1919 film *The Cabinet of Dr. Caligari* [64]. Many film directors have used a technique popularized by Hitchcock [32] that manipulates focal length in a non-realistic manner to create a feeling of uneasiness. In the animation world, Disney's multi-plane camera, a forerunner to modern compositing systems, allowed images painted from different of view to be layered and moved in such a way that shots appear to have an artificial appearance of spatial depth [59]. Another popular animation device is distortion glass, effectively a physical 2D distortion that has been used to mimic the appearance of flowing water, for example in the Nutcracker sequence of the film Fantasia [5].

More recently, digital manipulation has been used create the appearance of distorted perspective. Image warping, popularized by ILM in films such as *Willow* [34] and *Star Trek IV: The Voyage Home* [48], has seen a great deal of use, especially in experimental short films and music videos. Michel Gondry made excellent use of the technique in the Rolling Stones' "Like a Rolling Stone" to manipulate the timing of human motion in live-action footage [25]. More recently, multiple camera systems, as developed by Tim MacMillan [1] and Dayton Taylor [58], have been used to decouple the time base of a camera from that of the world. Now popularly known as the "bullet-time" effect, the technique has been used create dramatic shots with the appearance of implausible camera speed [62], and has also been extended to computer animation systems [2, 29].

2.2 Computer Graphics Projection

Projection is one of the fundamental operations within the graphics pipeline, and various types of projection are described in a number of texts. The first part of this section describes the qualities and limitations of projective transformations commonly used for creative purposes; the reader is referred to a text such as Foley et al. [20] for taxonomies of

projection and more detailed derivations. The remainder of this section reviews various approaches to incorporating projective nonlinearities and discontinuities in computer-generated images.

2.2.1 Linear Projection Models

Two types of linear projection are predominantly used in graphics systems. *Orthographic* projection maps three-dimensional geometry to an image by eliminating the depth component of a coordinate system centered at the viewer, with one axis aligned to the viewing direction (*eye* space). This nonrealistic projection has strengths in providing clear visual cues of relative size without perspective forshortening. As such, it has found a great deal of use in design and architectural modeling software. In naturalistic image synthesis, this projection is commonly done after performing a projective transformation that maps a volume of eye space to a canonical view volume. After this volume is projected, it is commonly mapped to screen space with a 2D translation and scale. Expressing homogeneous points as column vectors, the orthographic projection transformation of a right-handed eye-space rectangular viewing volume to a canonical view volume is

$$\mathbf{p}' = \begin{bmatrix} \frac{2}{R-L} & 0 & 0 & \frac{R+L}{L-R} \\ 0 & \frac{2}{T-B} & 0 & \frac{T+B}{B-T} \\ 0 & 0 & \frac{2}{N-F} & \frac{F+N}{N-F} \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{p}, \tag{2.1}$$

where L and R, B and T, and N and F represent the horizontal, vertical and view-vector aligned dimensions of the eye-space viewing volume, respectively. This transformation preserves parallel lines and relative lengths, and is useful for understanding dimensionality relationships. However, it does not create naturalistic imagery as each pixel can be interpreted as having its own unique viewing position.

The *Perspective* transformation used in naturalistic image synthesis maps a view *frustum* in eye space to the canonical view volume such that all points in the image

appear to be seen from a single viewpoint. This is commonly specified by explicitly providing the view volume dimensions L and R, B and T, and N and F. In this case, the horizontal and vertical dimensions are specified at the near distance N, and the resulting transformation is

$$\mathbf{p}' = \begin{bmatrix} \frac{2N}{R-L} & 0 & \frac{R+L}{R-L} & 0 \\ 0 & \frac{2N}{T-B} & \frac{T+B}{T-B} & 0 \\ 0 & 0 & \frac{F+N}{N-F} & \frac{2FN}{N-F} \\ 0 & 0 & -1 & 0 \end{bmatrix} \mathbf{p}. \tag{2.2}$$

These transformations bring visible space into a canonical view volume such that $x \in [-1, 1]$, $y \in [-1, 1]$ and $z \in [-1, 1]$. The depth component is discarded, and a two-dimensional viewport transformation brings the x and y components into image space by means of a translation and a scale [20]. This perspective transformation differs from real-world viewing in that parallel lines orthogonal to the viewing direction will not converge as distance from the center of projection increases.

Artists using traditional media typically develop perspective projections by specifying vanishing points for particular axes in space. Johnston [36] demonstrates how three world space axes $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$ with origin $\mathbf{o}$ can be transformed such that they converge to eye space vanishing points $\mathbf{p_a}$, $\mathbf{p_b}$, $\mathbf{p_c}$ and origin $\mathbf{p_o}$ with the following weighted change-of-basis transformation:

$$\mathbf{p}' = \begin{bmatrix} a_x a_w & b_x b_w & c_x c_w & p_{ox} p_{ow} \\ a_y a_w & b_y b_w & c_y c_w & p_{oy} p_{ow} \\ a_z a_w & b_z b_w & c_z a_w & p_{oz} p_{ow} \\ a_w & b_w & c_w & p_{ow} \end{bmatrix} \mathbf{p} \tag{2.3}$$

This is then followed by the above orthographic transformation to bring points into the canonical view volume. The weight entries a_w, b_w, c_w, and p_{ow} manipulate the rate of perspective foreshortening along the individual axes, or overall with respect to the origin.

2.2.2 Image Warping

Image warping techniques manipulate the final appearance of an image by defining a two-dimensional distortion that is used to sample an undistorted source image. Early techniques were based on deforming spline grids, and are described by Wolberg [65]. Max introduced a technique that warps images for off-axis projections and projections onto curved surfaces [46]. Dorsey et al. used projected grids to warp source images for realistic image projection onto arbitrary scene elements in an opera setting [17]. A feature-based technique was introduced by Beier and Neely in which the transformation of feature proxies defined on image pairs is used to construct a warp for morphing applications [8]. The combination of the effectiveness of their approach and its ability to allow specification of direct correspondence among features has led to its widespread use in commercial applications. Lee et al. build upon both the spline grid and feature-based approaches, using snakes as feature proxies in conjunction with multi-level free form deformations to generate warps [42]. Zorin and Barr [71] introduce a two-dimensional transformation technique that can be combined with either a perspective or parallel projection to correct for the perceptual distortion of viewing transformations. View morphing [55] extends image morphing to incorporate qualities of viewpoint change to create morphs with compelling three-dimensional appearances. Yang et al. [68] create projections onto cylindrical and spherical image surfaces by calculating equivalent warp functions that are applied to images generated with perspective projection. While many of these techniques offer expressive power in the image manipulation domain, they are not designed for applications where 3D models are the primary form of data. When the stylized rendering of 3D scenes is the primary application (for example, Yang et al.), the technique is inherently limited to producing images with the same composition and occlusion relationships present in the source images.

2.2.3 Continuous Multiperspective Mosaics

Continuous multiperspective mosaics are created by incorporating information from either conventionally captured or rendered images into a single image with a continuously changing viewpoint and/or view direction, also known as *pushbroom images* [30]. A simple form of such mosaics is created by rotating a camera in place and sampling a vertical strip of pixels for each camera view. These are horizontally composited to create an image in which the view direction is continuous with respect to the vertical direction but continuously changes along the horizontal direction. Rademacher and Bishop [51] extend this technique by also varying camera position to create source images for image-based rendering. They constrain adjacent camera views to horizontal translations and pans; this retains the single view position and direction within any vertical strip of the resulting image (often referred to as a *panoramic mosaic*). Shum and He use a set of *concentric mosaics*—panoramic mosaics captured from a set of cameras attached to a rotating arm—as a database for image-based rendering [56]. Peleg et al. describe a technique that allows arbitrary continuous camera motion to be used in the creation of a multiperspective mosaic [49]. They sample strips of the captured image and reproject them to a manifold that adapts, based on the camera motion, to approximate the appearance of panoramic mosaics. More recently, a formal characterization of stereo views has led to the use of multiperspective imagery as a data source for stereo vision algorithms [53, 54].

Inspired by their use in traditional cel animation, Wood et al. create a set of multiperspective images for an animated camera in a 3D scene [66]. A resulting layered set of large images can be illustrated in great detail and used in conjunction with a multiplane camera to approximate the appearance of 3D camera motion using the 2D medium. These techniques, along with the above mosaicing approaches, are primarily intended as underlying systems for other applications. Their continuously changing perspectives do not adapt well to ithe interactive rendering of 3D scenes.

2.2.4 Nonlinear Ray Tracing

Nonlinear ray tracing extends conventional ray tracing by defining camera rays that sample a scene such that the resulting image is not equivalent to a linear perspective projection. An early technique by Wyvill and McNaughton modeled viewing as an *optical model*, in which the camera rays are defined as mathematical functions with respect to image space [67]. They demonstrate two approaches to creating such models: one method modifies the view directions alone to simulate fish-eye lens effects, and the other modifies both view positions and directions to follow an analytic surface in space. Bourgoin et al. also define rays as nonlinear function of space, modeling a variety of perspective deviations common in Western art [10]. The approach to defining rays as functions of surfaces in space was extended to parametric surfaces by both Johnston [36] and Glassner [22]. Pixels are linearly mapped to the surface's parametric coordinates, and the ray origins and directions are constructed from the surface location and normal vector. Löffelmann and Gröller consider mappings of pixels to superquadric surfaces for their abstract camera models [44]. Again, rays are defined with respect to the surface geometry. Vallance and Calder develop an OpenGL-like API for specifying such a surface with independent local control of ray directions [61]. Kolb et al. are able to recreate the geometric nonlinearities of real camera projections by tracing rays through models of physical lens systems [40]. Gröller takes a different approach, in which ray paths are warped in space as though affected by massive objects or dynamical attractors [28]. Weiskopf et al. adapt this approach to the GPU computation model [63]. While nonlinear ray tracing offers the most general set of views, including the ability to visualize a point at multiple image locations, the computational expense of the technique precludes interactive projection manipulation.

2.2.5 Multiple View Projections

In this thesis, we refer to *multiple view projections* as techniques that incorporate a discrete set of views to project scene geometry. This is in contrast to nonlinear ray tracing and multiperspective images, in which the viewpoint definition can change continuously with respect to one or both image dimensions.

Levene [43] describes a framework for creating a particular class of nonlinear projections. These projections are defined as radial deformations of scene geometry with respect to eye space. Different nonlinear projections can be applied to different groups of objects, although each projection is defined in terms of a single view. While this approach introduces distorted projections, different regions of a scene can not be projected relative to different views. Agrawala et al. [3] describe an approach to rendering objects from different views and interactively compositing them together to construct a multiprojective image. Each object is seen from a unique linear perspective camera. Visibility is resolved using a combination of depth compositing with respect to a single *master* camera and a set of intra-object occlusion constraints. They also incorporate techniques to position individual cameras using constraints on object position, size, and orientation. Grimm [27] presents a similar approach, although compositing is an isolated post-process of individually rendered objects, and partial occlusion relationships are not addressed. The approaches developed by Agrawala et al. and Grimm can be used to incorporate multiple linear perspective projections into in image, but they do not allow for continuous changes in vantage point.

Another approach allows the user to directly lay out rendered image fragments onto a common image plane to locally express desired views and an overall composition. Yu and McMillan [69] present a system in which the camera associated with each image fragment is parameterized by three rays (*general linear cameras*). Rays are blended in regions of fragment overlap or among nearby fragments where the image plane is not covered. Adjaceny constraints are enforced among the views over a small-scale image

tesselation to maintain ray-space continuity with respect to the image plane, although this approach does not allow for the viewpoint dicontinuities common in traditional artwork with multiple projections. Glassner describes a system with a similar interface, but details await further publication [23].

Related to these techniques are approaches to stylized rendering that incorporate multiple views from pre-existing images into composite renderings. Video Cubes [39] allow different views from within a video sequence to be displayed in spatial partitions of a resulting video. Video mosaics are similar in this regard, but they are constrained to a grid partition and the different views are used to create a video sequence whose large-scale appearance appears similar to another video [38]. The Cubist rendering technique of Collomosse and Hall [16] statistically analyzes a set of images from multiple views to determine feature relationships. Portions of these images are then combined into a composite image depicting an object from multiple views while maintaining feature correspondence among adjacent image fragments. Painterly rendering techniques are applied to further stylize the resulting images.

Singh [57] introduces the idea of using multiple linear perspective cameras to create and control nonlinear projections. Any location in the scene is associated with a linear combination of these cameras' views, where the weight of each camera is defined as a function in space related to either specific spatial locations or the view vectors of the various cameras. Near to these locations or centers of view, the camera has full weight, which continuously falls off to zero at some threshold distance. Scene geometry is projected using a virtual linear perspective camera that is defined from weighted interpolations of the multiple cameras' parameters. This geometry is rendered onto regions of a common canvas, and visibility is resolved using a combination of the local cameras' projected depth values and per-camera depth offsets. The nonlinear projection approach described in Chapter 3 of this thesis builds upon Singh's idea of using multiple linear perspective projections, but uses interpolated projections in place of per-vertex virtual

cameras. This avoids the need to construct projection matrices independently for each point, significantly reducing computational overhead. It also incorporates the projection technique into a comprehensive rendering pipeline that can be incorporated into existing graphics systems, and it allows animators to work with an animated camera within the scene.

Another approach to nonlinear projection was introduced by Chu and Tai [14]. In their work, they create animated fly-throughs of Chinese landscape paintings by combining linear approximations to local regions of the painting through the use of billboards and texture projections to enclosing boxes. At a large scale, the painting is projected to cylindrical or spherical surfaces, in which the local linear views are embedded. Smooth interpolations between the projections are used to create fly-through animations of the large scale panoramic projection characteristic of the paintings.

2.3 Manipulating Projection for Information Visualization

The information visualization and interaction communities have conducted a great deal of research into incorporating multiple views into a conventional display frame. The primary motivation for these techniques is the presentation in detail of important information while retaining context information in less detail. Carpendale [12] provides an excellent overview of such techniques; this section provides a brief review. *Windowing*, the technique underlying popular graphical user interfaces, partitions information into a discrete set of views which can be tiled, overlapped, or replaced by icons within a larger abstract view. Individual views are zoomed or scrolled to locally explore information. *Bifocal displays* show important information in a zoomed presentation, and context information is compressed or warped to fit into the remaining presentation space. *Fisheyes* extend this idea by gradually decreasing the detail with increasing distance from impor-

tant regions. This approach also allows for the incorporation of multiple detail regions. A variety of 3D techniques are also discussed with the primary aim of creating detail-in-context presentations. Carpendale goes on to describe the *elastic presentation framework*, which encapsulates all of these techniques within a common framework where specific presentations are described as lenses [12, 11]. This framework describes the presentation of 2D information as a display on a planar surface within a perspective projection. This surface is then manipulated in space to achieve a desired presentation [11].

2.4 View Dependent Deformation

Deformation was originally introduced as a high-level modeling tool, but has since been applied to many problems, including character rigging and the animation of secondary motion. Barr [6] expresses deformations as a hierarchy of transformations applied to object geometry to construct complex modeling tools. Sederberg and Parry [52] introduced the popular approach of using control point lattices to define a deformation volume for embedded geometry. Camera rays were deformed by Barr [7] to more efficiently ray trace deformed parametric surfaces. This technique was adopted in Gröller's nonlinear ray tracing approach [28], described earlier. Camera sensitive deformations of geometry were applied by Rademacher [50] to create models that deform to shapes more appropriate for particular views, an idea inspired by the view dependent shapes of traditionally animated characters. This idea was also applied by Martín et al. [45]. These techniques are related to the nonlinear projection approach described in Chapter 3 in that the projections are accomplished with camera-sensitive deformations of scene geometry that produce the desired image from a particular view.

2.5 Controlling Cameras and Projections

Chapters 4 and 5 of this thesis introduce techniques for the computational control of multiple cameras used to define a nonlinear scene projection. Two common approaches for controlling cameras in space are direct specification of either transformation matrices or a combination of location, view direction, and tilt [20]. Projections are typically specified with parameters analogous to real-world camera parameters such as focal length, angle of view, and aspect ratio. Blinn introduced a technique that directly solves for camera parameters given a specific, limited set of composition constraints [9]. Gleicher and Witkin used constrained optimization to solve for temporal derivatives of camera parameters, given a set of image composition constraints and the temporal derivatives of any fixed set of camera control parameters [24]. Gooch et al. [26] use a downhill simplex solver to generate camera parameters for a view that is considered aesthetically pleasing, given heuristics for ideal aspect ratios, viewpoints, and compositional layouts. The automatic camera-layout technique described in Chapter 4 for modeling nonlinear projections is similar to these approaches in that a given set of image composition constraints are used to derive a nonlinear projection.

The procedural control of an animated camera has been investigated by both the computer graphics and artificial intelligence communities. These techniques attempt to model camera motion and cuts using principles of cinematography [37]. Drucker et al. developed a system in which camera motion primitives such as *zoom to* and *track* are built into a set of *frameworks*, each of which describes a common cinematic sequence [18]. Christianson et al. formalize camera motion primitives as a language which can be used to author *film idioms*, which are analogous to Drucker's frameworks [13]. A finite state machine was used by He et al. to describe these film idioms and control the transitions among motion primitives [31]. Camera control is cast a cognitive modeling problem by Funge et al. [21]; camera motion in an animated scene was described in a cognitive modeling language developed for behavioral character animation. Tomlinson et al. describe

a behavioral approach to camera control in which camera motion primitives are tied to an ethologically inspired emotional state. Action selection mechanisms are used to transition among these states [60]. The procedural camera control models developed in Chapter 5 have similar motivations as these techniques. The approaches described above are complementary in that they procedurally control a single camera rather than a set of cameras that define a nonlinear projection.

Chapter 3

Nonlinear Projection of Complex Animated Scenes

The creation of general nonlinear projections for computer-generated imagery presents a variety of technical challenges. In particular, many nonlinear projection techniques are not amenable to interactive manipulation or do not scale well to complex scenes. This chapter presents a system for creating and rendering nonlinear projections that is appropriate for use in conventional animation pipelines. In particular, projection is accomplished as a geometric deformation relative to a conventionally-animated camera. Projective manipulations are added incrementally as desired, and geometry can be associated with one or more linear perspective cameras. A constraint system is incorporated whereby scene composition can be locally controlled. The effect of the nonlinear scene projections on rendering is addressed, and two approaches to incorporating the multiple camera views into the illumination model are presented. The system has been used in the production of an animated film, *Ryan*, validating its usefulness and creative power.

The next section of this chapter describes the characteristics of the system and the concepts associated with its use. Section 3.2 describes in detail the projection model and

[1]This chapter presents work with Karan Singh published in [15]

constraint formulation. In Section 3.3, rendering issues are addressed, and the incorporation of multiple views into illumination calculations is discussed. The implementation of this model in the animation system *Maya* is presented in Section 3.4, and Section 3.5 presents selected results.

3.1 Creating a Nonlinear Projection

In developing our model for nonlinear projection, a wide variety of qualities have to be considered. Perhaps the most difficult design issue is allowing users to work with the system to achieve intuitive results. Users should have the ability to create a wide variety of projection styles, but should not be concerned with underlying technical details. As the system is intended to work in a production environment, it should robustly and interactively handle the complex scenes animators typically work with. It should support a wide variety of geometric primitives, and should incorporate a coherent rendering pipeline. Finally, the user should be able to easily edit projection at both a global and local level, and he should be able to directly create desired compositions. We now provide a high-level description of the system, and then consider how its design meets the given criteria.

A conventional 3D animation workflow uses a single linear perspective camera for setting up and animating a shot; we refer to this as the *master* camera. *Lackey* cameras are added as necessary to represent different target linear perspective views. Each lackey camera can be associated with a subset of scene geometry, which will appear in an image rendered with the master camera as though seen by the lackey camera, allowing for the incorporation of multiple projections within an image. To incorporate continuously-varying projection effects, different local regions of a scene can be associated with different cameras. The remainder of the scene will appear as a smoothly varying projection. Cameras are associated with corresponding regions of a scene through the use of weight functions

over space. Individual cameras can have greater influence in important regions, for example along lines of sight and near specific points of interest. This continuously varying perspective is evident in the pre-production artwork from *Ryan* shown in Figure 3.1. In this rendering, multiple nonlinear deformations are used to illustrate the desired style.

In addition to having local control over the projection of scene elements, an animator should be able to independently manipulate the composition of the scene. General 2D transformations should be supported, including changes to the translation, rotation, and scale of objects.

Control over the illumination and shading of the scene is defined in terms of the rendering parameters associated with the master and lackey cameras. Conventional illumination as seen from the position of the master camera does not incorporate view-dependent effects that would appear differently in the lackey camera views. This results in an appearance discrepancy between the local regions of the nonlinearly projected image and the linear perspective projections of the lackey cameras used by the animator to define the nonlinear projection. Two methods for incorporating the multiple views into illumination calculations are introduced—the use of an interpolated camera view and the blending of illumination from the multiple view directions (Figure 3.7). The interpolated view model offers no unique stylistic deviations in illumination, but the blended illumination model allows for greater creative exploration. Furthermore, it is more predictable as the final illumination is a composite of that seen from the lackey camera linear perspective views.

In designing the nonlinear projection system described in this chapter, both an understandable, easy-to-use technical approach and an intuitive workflow had to be developed. After consulting with experienced animators who would be using the system in production, a set of design criteria was put together to assist in development. We now list these criteria, and describe how the system addresses these challenges.

- **Local linear perspective**: Each of the cameras defines a linear perspective view.

The overall scene is a blend among these cameras' views. To maintain local linear perspective, the weight functions are constructed such that they smoothly vary in space.

- **Global scene coherence**: Scene coherence is necessary, as a certain degree of linear perspective spatial relationships is necessary to provide a viewer with an understandable depiction of 3D relationships among objects. Constraints were designed primarily for this purpose, as they allow users to tie specific geometry to specific location in the image and to manipulate relative depth relationships among objects.

- **Conventionally animated underlying view**: Projection is accomplished as a world-space deformation such that the master camera's projection of the deformed 3D scene results in the appropriate nonlinear perspective view. This allows animators to specify motion for the master camera that causes it to move through the scene as a conventionally animated camera.

- **Default linear perspective**: Nonlinear projections typically have an underlying linear perspective quality, with nonlinearities affecting the projection is often subtle and continuous ways. To this end, in the absence of any lackey cameras, the scene's projection in the linear perspective projection of the master camera. Each of the lackey cameras also has an associated linear perspective view, with the resulting projection being a blend of linear perspective projections.

- **Local control of vantage point**: Lackey cameras' regions of influence are defined relative to given features and directions of view, and their associated weight functions fall off with spatial distance. This allows users to manipulate projection at a local level by working with a camera that affects a limited region of space.

- **Ability to incorporate multiple projections**: Each lackey camera can be as-

sociated with a subset of the objects in a scene. This allows for the creation of projections that incorporate a discrete set of linear perspective views. More generally, a set of cameras can be associated with a group of objects, allowing for multiple nonlinear projections to be incorporated into an image.

- **Independence of composition and projection**: To achieve a desired projection, a camera may need to be positioned such that an object appears far from a desired location in image space. Viewport transformations and constraints allow the user to change the location and size of objects without changing the projection, and hence appearance, of various scene elements.

- **Global projection manipulation**: The viewport transformation modifies the projective qualities of a given camera throughout the scene for all objects which it affects. Thus, the user can transform all geometry associated with a given camera as a whole.

- **Local projection manipulation**: Constraints act similarly to the viewport, but can be created at a local level and blended in space when multiple constraints are associated with a given view.

- **Control of relative depth relationships**: The viewport transformation is defined such that offsets in depth can be defined globally on a per camera basis. The user can take advantage of this feature to bring an entire camera's view forward or back in space without changing the image composition or projection. Constraints offer the same power, but can be defined at a local level to allow for local manipulations of depth relationships.

- **Coherent rendering pipeline**: The rendering pipeline compensates for the scene deformation to avoiding creeping shadows and illumination artifacts that arise from the deformation approach. In addition, the multiple views are incorporated into a

blended illumination model.

- **Incremental work-flow**: The user initially works with only the master camera. Lackey cameras are introduced as desired to manipulate projection. When lackey cameras are introduced, they are set to have the same parameter values of the master camera, thus not changing the projection until the user edits the camera. A default location constraint is also introduced to the geometry affected by that camera to constrain it relative to a central point. This is desirable, as most camera edits will significantly change the image composition, which is typically not desired.

- **Interactive manipulation of complex scenes**: To handle complex scenes, the nonlinear projection model is formulated to allow for maximum pre-computation. Given the parameter values of all cameras, the set of constraints, and viewport transformations, a single transformation matrix is computed for each camera. The nonlinear projection of a point involves one matrix multiplication for each camera and the computation of the spatial weight function at the point's location.

3.2 Nonlinear Projection Model

The idea of constructing nonlinear projections as a combination of linear perspective views was first introduced by Singh [57]. In his approach, the viewports of a set of exploratory cameras are laid out onto a common rendering canvas, and each camera influences regions of a scene based on local weight function values. The projection model described here uses the master and lackey cameras in place of his exploratory cameras, and projection is effected by scene deformation rather than as an integral part of the rendering process. The weight functions of the cameras in Singh's approach are computed based on distances from either the view vector of a given camera or a position in space. This approach to defining weight functions is adapted from his approach, as it allows for

Figure 3.1: A pre-production composite of multiple nonlinear deformations. The deformations of the pillars, walls, and figure illustrate qualities of continuous variation in perspective, while maintaining the spatial appearance of the underlying linear perspective view.

easy specification of relevant content. Composition in our technique is controlled through the use of constraints and manipulations to the viewport transformation, rather than direct control of rendering layout. In addition, projection can be selectively activated either globally or per-camera, and it can be attenuated based on deformation magnitude.

3.2.1 The Camera Transformations

Assume points have a homogeneous representation such that a world space point $\mathbf{p}_w =< x_w, y_w, z_w >$ is represented as a four-tuple $< x, y, z, w >$, where $\mathbf{p}_w =< x/w, y/w, z/w >$ after a perspective division. Consider the viewing, perspective projection, and viewport transformations, $\mathbf{E}$, $\mathbf{P}$, and $\mathbf{V}$ respectively. $\mathbf{E}$ is a 4 x 4 transformation matrix representing the viewing transform of the camera (see Chapter 2). $\mathbf{P}$ is the 4 x 4 perspective transformation matrix that linearly warps space, such that after a perspective division, it appears as a linear perspective projection (Chapter 2). $\mathbf{V}$ is a viewport transformation,

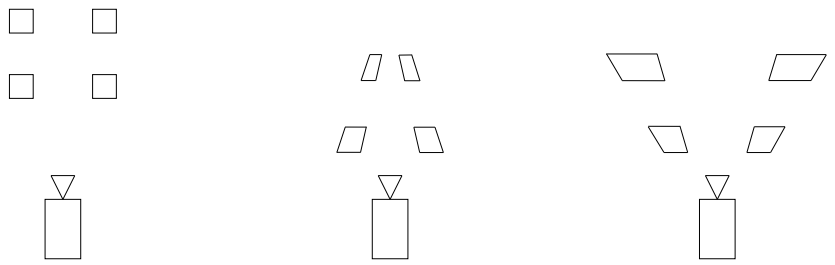
represented as a 4 x 4 matrix that maps the x and y components to image space from a canonical view volume. For notational simplicity in the following discussion, these are encapsulated as a single camera transformation $\mathbf{C} = \mathbf{VPE}$. $\mathbf{C}_m$ denotes the camera transformation of the master camera, and $\mathbf{C}_i$ represents the camera transformation of lackey camera i, where $i \in [1, n]$ is a labeling of n lackey cameras. $\mathbf{p}_c = \mathbf{PEp}_w$ is then a linear perspective transformation of a world-space point $\mathbf{p}_w$ into a camera's canonical view volume such that visible elements map to $x_c \in [-1, 1],\ y_c \in [-1, 1],\ z_c \in [-1, 1]$ after perspective division.

Typically, in graphics systems, a perspective division is carried out after the perspective transformation $\mathbf{P}$ is applied. The viewport transformation $\mathbf{V}$ then translates and scales the x and y coordinates into image space. We instead incorporate a user-controllable viewport transformation; we now have image space points (that retain depth) $< x_s, y_s, z_s >= \mathbf{Vp}_c$. By default, $\mathbf{V}$ is an identity transform; introducing translation or scale components to this matrix allows the user to control the composition and relative depth relationships among the views of various cameras. Within a single view, however, depth relationships are preserved. Note that this viewport transformation is carried out without perspective division. Therefore, the magnitude is sensitive to the focal length of the camera. While this is slightly non-intuitive to the user, it allows for the camera transformations to be pre-computed, necessary for efficiently working with complex scenes. Furthermore, the interactive nature of the systems allows users to achieve predictable results when manipulating this transformation.

3.2.2 Inverse Projection Matrices

The nonlinear perspective system descried in this chapter relies on the invertibility of the perspective projection matrix. By nature, true projection matrices are not invertible. However, perspective projection, as commonly implemented in graphics systems, makes use of a perspective *transformation* that maps world space points to a canonical view

volume. Additionally, after projection, points exist as a homogeneous representation in four dimensional space.



(a) (b) (c)

Figure 3.2: Geometric effect of inverse perspective. a) A top down orthographic view of the 2D scene, with camera at bottom. b) In the normal perspective transformation, close objects appear greater in size to a viewer centered at the camera, modeling perspective foreshortening. c) Inverse perspective transformations model inverse perspective, where distant objects appear greater in size to the viewer.

The perspective transformation, commonly referred to as the projection matrix [1], is of full rank and hence, invertible. This can be shown by analytically deriving the inverse of the perspective transformation from Chapter 2. This inverse is:

$$\mathbf{P}^{-1} = \begin{bmatrix} \frac{R-L}{2N} & 0 & 0 & \frac{R+L}{2N} \\ 0 & \frac{T-B}{2N} & 0 & \frac{T+B}{2N} \\ 0 & 0 & 0 & -1 \\ 0 & 0 & \frac{N-F}{2FN} & \frac{F+N}{2FN} \end{bmatrix} \tag{3.1}$$

[1]The perspective transformation, whose underlying matrix is commonly referred to as the projection matrix, is not a true projection matrix. We retain the use of the generic term *projection matrix*, however, as we do not require the transformation to a perspective transformation. It must, however, be an invertible mapping of an eye space view volume to the canonical view volume.

This matrix warps a canonical view volume into the viewing frustum associated with the viewing parameters used to construct the matrix (see Figure 3.2). In particular, any point previously projected with the associated perspective projection transformation will have the w component restored to one, and the original values of the remaining coordinates will be restored.

3.2.3 Combining Multiple Linear Projections

Scene deformations are computed such that only the master camera views a nonlinear projection. First we describe how to deform a world space point $\mathbf{p}_w$ such that it appears in the master camera's view as though seen from a lackey camera (Equation 3.2). We then build upon this to describe how to interpolate among the views of multiple lackey cameras (Equation 3.15). This approach is then reformulated to allow for the accumulation of deformation effects on a per-camera basis, enforcing that the weight values sum to one (Equation 3.16). We do not enforce that weight functions must be positive, as non-convex combinations also produce pleasing results.

The goal is to deform a world-space point $\mathbf{p}_w$, such that when viewed through the master camera, it appears as though viewed by the ith lackey camera. For a single undeformed world-space point, all points that project to the desired image point (which create a line in 3D) will meet this goal. We wish for the deformation transformation to choose a single point, such that nearby points will be deformed in a manner that preserves depth relationships with respect to the master camera. We use the following transformation:

$$\mathbf{p}'_w = \mathbf{C}_m^{-1}\mathbf{C}_i\mathbf{p}_w \tag{3.2}$$

The result of the transformation is a Cartesian point that maps to a unique location along the line of points that project to the desired point in the image. Relative depth relationships are preserved, as the transformation preserves relative magnitudes of pseudo-depth. In effect, this transformation replicates the view of lackey camera i in the master camera

by deforming all points in space.

Multiple, per-object, linear perspective projections can be combined by selectively deforming objects using the single set of matrices for the master camera, and, for each object, the matrices of the associated lackey camera. Each object is deformed using the above deformation. To control relative depth values among objects that overlap, the viewport transformation is extended to affect depth values, in addition to vertical and horizontal translation and scale. Thus, the user can interactively translate and scale the deformed geometry in depth alone, allowing him to compose an image with desired occlusion relationships.

As described in the previous section, if the perspective matrices of the master camera and a lackey camera are equal, the projection is effectively undone by the master camera's inverse projection matrix, and the eye space point is restored. However, it is worth investigating how differing values for the camera projection parameters affect the deformation. In the following analysis, we consider two variations of the perspective transformation. First, we consider a simpler variant of the projection matrix from Chapter 2. This variation assumes a centralized projection with equal height and width, such that $L = -R = T = -B$. The constant focal length f represents the inverse scale applied to the near clipping plane for a centralized perspective projection. The resulting matrix is as follows:

$$\mathbf{P} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & \frac{F+N}{N-F} & \frac{2FN}{N-F} \\ 0 & 0 & -1 & 0 \end{bmatrix} \tag{3.3}$$

$$\mathbf{P}^{-1} = \begin{bmatrix} \frac{1}{f} & 0 & 0 & 0 \\ 0 & \frac{1}{f} & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & \frac{N-F}{2FN} & \frac{F+N}{2FN} \end{bmatrix} \tag{3.4}$$

Changing the value of the focal length for a lackey camera has the same effect as changing the focal length of a standard camera that views undeformed geometry. Increasing the focal length of the lackey camera will expand the size of an object in the final image rendered by the master camera. To demonstrate this, consider both a master camera and a lackey camera at the origin, oriented and scaled such that their transformation matrices are the identity matrix. Also assume identity matrices for the viewport transformations. While this analysis is general, the expressions resulting from matrix multiplication are more intuitive when eye space and world space are equivalent. the deformation we are interested in analyzing is as follows:

$$\mathbf{p}' = \mathbf{P}_m^{-1}\mathbf{P}_l\mathbf{p} \tag{3.5}$$

This multiplication represents the deformation of a master camera eye space point $\mathbf{p} =< x, y, z, w >=< x, y, z, 1 >$ to the deformed master camera eye space point $\mathbf{p}' =< x', y', z', w' >$. When both the master camera and lackey camera have the same value for a perspective transformation parameter (for example the focal length f), this value will be denoted with with no subscript. If the master camera and lackey camera have different values for a parameter, we append the subscripts m and l to differentiate these values (for example f_m and f_l).

Carrying out this deformation multiplication for master camera focal length f_m and lackey camera focal length f_l, the x and y components will be changed, but z will not and w will remain one. The expressions for the altered components of $\mathbf{p}'$ are as follows:

$$x' = (f_l/f_m)x \tag{3.6}$$

$$y' = (f_l/f_m)y \tag{3.7}$$

Thus, if the lackey camera has larger focal length than the master camera, the master camera will view an enlarged image, equivalent to that it would view if no lackey camera were present and its own focal length had been increased.

If each camera has the same values for N and F, w' will be one. The remaining components will be unchanged. The expression for w' is as follows:

$$w' = (2N_mF_l - 2N_lF_m)z + 2F_lN_l(N_m - F_m)/(2F_mN_m(N_l - F_l)) \tag{3.8}$$

After dividing $\mathbf{p}'$ by w', this expression represents a translation and a depth-dependent scale with respect to the original depth value z. It can be shown that this, overall, is equivalent to a uniform scale in depth alone within the canonical view volume. Because of this linearity within the canonical view volume, changes to either the near or far cutting plane of the lackey camera with respect to those of the master camera will not change positional relationships within a rendered image. Changing either of these values effectively translates and scales the space contained within the lackey camera's viewing frustum such that it fits exactly within the frustum of the master camera. While image position is not changed, occlusion relationships can be changed when these parameters are manipulated. However, as near and far cutting planes are typically used to control the region of space being rendered, we do not expose them as controls for manipulating occlusion relationships. In particular, we constrain the near and far cutting planes of all lackey cameras to be equal to those of the master camera, preventing changes in occlusion relationships when these values need to be manipulated for the master camera.

We now consider the deformation of a point $\mathbf{p}$ using the general perspective transformation matrix from Chapter 2, whose inverse is given above. We use the same notational

conventions described above for the special case of central projection. The free parameters we are interested in are N and F, the distances to the near and far cutting planes, L and R, the signed distances on the near plane to the horizontal edges of the viewing frustum, and T and B, the signed distances to the top and bottom edges on the near cutting plane to the vertical edges of the viewing frustum.

Consider changing only N. The deformed point $\mathbf{p}' = \mathbf{P}_m^{-1}\mathbf{P}_l\mathbf{p}$ can be expressed as follows:

$$\mathbf{p}' = \begin{bmatrix} \frac{N_l}{N_m}x \\ \frac{N_l}{N_m}y \\ z \\ \frac{N_l(N_m-F)}{N_m(N_l-F)} + \frac{N_m-N_l}{N_m(N_l-F)}z \end{bmatrix} \tag{3.9}$$

Consider an increase in N_l. The near plane of the lackey camera is now more distant, so the space within its viewing frustum must be expanded horizontally and vertically to fill the master camera's viewing frustum, with magnitude decreasing with depth. The expressions for x' and y' are indicative of this horizontal and vertical expansion. The expression for w' accounts for the change in perspective foreshortening. It accounts for the decreasing magnitude of horizontal expansion as z increases. It also accounts the the expansion in depth required for the lackey camera viewing frustum to fill the deeper master camera viewing frustum.

Now consider a change in F alone. The result of deformation is as follows:

$$\mathbf{p}' = \begin{bmatrix} x \\ y \\ z \\ \frac{(F_l-F_m)z+F_l(N-F_m)}{F_m(N-F_l)} \end{bmatrix} \tag{3.10}$$

Consider an increase in F_l. In this case, the field of view has not changed. Hence x and y do not have any uniform scale applied to them. However, perspective foreshortening

has changed, and the lackey camera viewing frustum must be compressed to fit into the master camera viewing frustum. The expression for w' accounts for this change in perspective foreshortening.

Considering a change in L, the deformation is as follows:

$$\mathbf{p}' = \begin{bmatrix} \frac{R-L_m}{R-L_l}x + \frac{R(L_l-L_m)}{N(R-L_l)}z \\ y \\ z \\ 1 \end{bmatrix} \tag{3.11}$$

A decrease in the magnitude of L_l (corresponding to widening the left half of the lackey camera's viewing frustum), would require the deformation to accomplish a uniform horizontal scale and a horizontal translation that increases with depth. This is required to align and compress the now wider lackey camera viewing frustum within the viewing frustum of the master camera. The first term in the expression for x' accounts for this uniform horizontal scale. The second term accounts for the horizontal translation that increases with depth.

When changing R, the deformation is as follows:

$$\mathbf{p}' = \begin{bmatrix} \frac{R_m-L}{R_l-L}x + \frac{L(R_m-R_l)}{N(R_l-L)}z \\ y \\ z \\ 1 \end{bmatrix} \tag{3.12}$$

Increasing R_l again widens the viewing frustum of the lackey camera, this time to the right. We again see the uniform scale in the first term in the expression for x', as well as the horizontal translation that increases with depth in the second term.

The deformation for a change in T is:

$$\mathbf{p}' = \begin{bmatrix} x \\ \frac{T_m - B}{T_l - B} y + \frac{B(T_m - T_l)}{N(T_l - B)} z \\ z \\ 1 \end{bmatrix} \tag{3.13}$$

And the deformation for a change in B is:

$$\mathbf{p}' = \begin{bmatrix} x \\ \frac{T - B_m}{T - B_l} y + \frac{T(B_l - B_m)}{N(T - B_l)} z \\ z \\ 1 \end{bmatrix} \tag{3.14}$$

As with the deformations that account for changes in L and R, these deformations account for a change in field of view. In particular, a decrease in B_l or an increase in T_l vertically expand the viewing frustum of the lackey camera, as the deformations must consist of a translation that increases with depth and an expansion or compression that is uniform. As with the expressions for x' when L or R was changed, the expressions for y' in these deformations consist two terms, one accounting . However, the uniform scale and translation that increases with depth are applied to the y component alone, as would be expected from a vertical widening of the lackey camera viewing frustum.

3.2.4 Nonlinear Perspective with Multiple Cameras

In general, lackey camera i only partially influences the point $\mathbf{p}_w$ based on a weight function $w_i(\mathbf{p}_w)$. The deformed world space point can then be expressed as a linear combination of the points as they would be deformed by each camera:

$$\mathbf{p}'_w = \sum_i w_i(\mathbf{p}_w) \mathbf{C}_m^{-1} \mathbf{C}_i \mathbf{p}_w \tag{3.15}$$

When combining the lackey cameras, the default view (each w_i is zero) is the linear perspective view of the master camera. The above form does not achieve this; it instead

defaults to collapsing all points to the origin. The desired affect of the master camera can be incorporated by treating it as an additional lackey camera. This however, leads to unnecessary deformation computation. Recognizing that the deformation due to the master camera is the identity transformation, and calculating its weight value such that $w_m(\mathbf{p}) = 1 - \sum_i w_i(\mathbf{p})$, we can plug these terms into Equation 3.15 and rearrange to achieve the following form:

$$\mathbf{p}'_w = \mathbf{p}_w + \sum_i w_i(\mathbf{p}_w)(\mathbf{A}_i - \mathbf{I})\mathbf{p}_w. \tag{3.16}$$

In this expression, $\mathbf{A}_i = \mathbf{C}_m^{-1}\mathbf{C}_i$ is substituted to simplify notation. The deformed point $\mathbf{p}'_w$ is now expressed as a relative offset, and the deformation of the point due to each camera can be progressively accumulated.

This accumulated deformation that models a linear interpolation among transformation matrices has been chosen for efficiency, as speed was of primary concern to handle the complexities of geometry used in production. Singh [57] advocates the construction of a virtual camera from interpolated parameters, suggesting that results will be more intuitive in the general case. However, the linear combination of matrices allows not only efficient calculation, but many terms of the deformation transformation can be precomputed. In practice, this method provides good results for all but extreme deformations; such cases can be handled by the use of chained lackey cameras (Section 3.2.7).

We now address three important issues relating to the control and usability of our nonlinear projection model: camera weight computation, constraints, and chained lackey cameras.

3.2.5 Camera Weight Functions

Figure 3.3 illustrates a number of spatial, per-camera weight functions introduced by Singh [57], which can be used to calculate the influence weights of cameras. A positional weight is computed as a radial falloff $f(s||\mathbf{p}_w - \mathbf{p}_i||^2 - t)$ from a center of interest $\mathbf{p}_i$. The

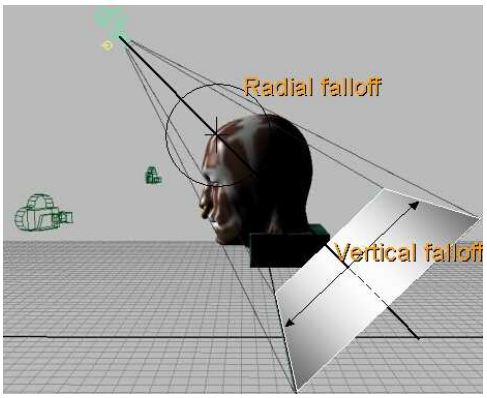


Figure 3.3: Camera weight computation (from Singh [57]). Points of interest can be assigned weight functions to a camera that depicts them as the user desires, and weight values fall off radially in 3D space. Regions close to the line of sight can be emphasized for a particular camera; in this case a horizontal region of emphasis along the camera horizon is shown.

nonnegative scalar t controls the size of a region with no falloff, and s, positive, scales the region of falloff. f should monotonically decrease from one to zero, have continuous first derivatives, and remain positive. We use a piecewise polynomial function:

$$f(x) = \begin{cases} 1 & \text{if } x < 0 \\ 2x^3 - 3x^2 + 1 & \text{if } 0 \leq x \leq 1 \\ 0 & \text{otherwise} \end{cases} \tag{3.17}$$

Vertical and horizontal falloff functions are defined within the camera's canonical view space relative to the x and y (horizontal and vertical) axes: $f(sx_c - t),\ f(sy_c - t)$. User-painted control point weights can also be specified explicitly on the geometry. Any other local attribute can be used; the surface normal, curvature, or any other analytic or computed property can be as important as surface position in determining a camera's influence on the surface. Figure 3.8 demonstrates the use of the facing ratio (the normalized angle between the surface normal and view vector) to assign weights to lackey cameras. In this image, lackey cameras are used to demonstrate an artificial rim lighting

effect from multiple viewpoints without distorting the scene geometry.

3.2.6 Constraints

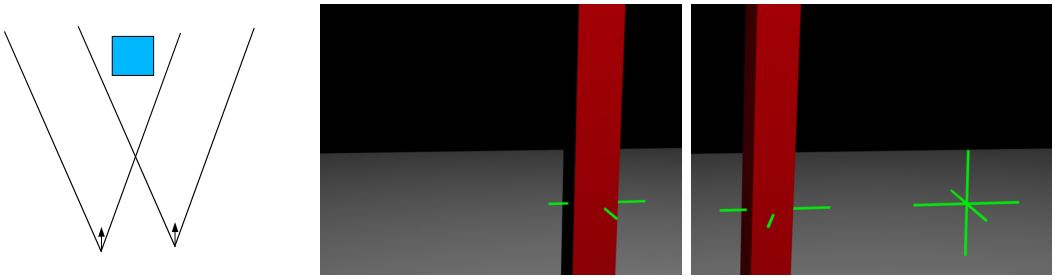
Constraints allow users to apply affine transformations to objects or regions of a scene while preserving the projective qualities defined by the master and lackey cameras. For example, in a multiple camera projection, it can be desirable to constrain objects in space to preserve their relative position and size in a composited scene. Agrawala et al. [3] address such constraints by manipulating the positions and orientations of the various cameras in space. Singh's approach [57] allows a user to control the relative position and size of a camera's projective effect through the use of a viewport transformation that maps exploratory camera views to rectangular regions of the rendering canvas.

Constraints are incorporated into our system to allow the user to locally specify the projection of the scene. In practice, they are typically used to constrain a group of objects relative to a particular location in image space. A single constraint can be used to force a particular point to project to a particular location. The user works with a pair of transformation reference frames in world space, represented with cross-hair widgets (Figure 3.4). The projection is modified such that the deformed world-space location of the *from* widget deforms to a point coincident in space with the world position of the *to* widget. More generally, we map the entire reference frame of the *from* widget, $\mathbf{R}_f$, to that of the *to* widget, $\mathbf{R}_t$, also incorporating effects of relative rotation and scale. Internally, we represent these reference frames with their world-space transformation matrices. In the following discussion, we use $\mathbf{R}_f$ and $\mathbf{R}_t$ to refer to both the widgets and their internal transformation matrix representation.

As an illustrative example, we wish to constrain a pillar in place, relative to a point in the final image, while changing its projective appearance. The undeformed pillar, as seen from a lackey camera in Figure 3.4b, would deform without constraints to the position of the widget to the right in Figure 3.4c, which is a view from the master

camera. The widgets are set up such that $\mathbf{R}_f$ is positioned at a point coincident with the location that $\mathbf{R}_t$ should deform to. $\mathbf{R}_f$ and $\mathbf{R}_t$ then define a transformation that undoes the deformation at the location of $\mathbf{R}_t$. The remainder of the geometry to which the constraint is applied undergoes the same uniform transformation. This results in the object having the lackey camera's projective appearance, but being "tied" in place relative to the location of $\mathbf{R}_t$. Without application of the constraint, the column would appear at the same location in screen space in each view, as the master camera would see exactly the lackey camera's view.

Mathematically, the goal is to have $\mathbf{R}_f$ deform to have the size, position, and orientation of $\mathbf{R}_t$, when viewed through the master camera. On a geometric level, points should deform such that their local coordinates, relative to $\mathbf{R}_f$, are mapped to equivalent local coordinates, relative to $\mathbf{R}_t$.



(a) (b) (c)

Figure 3.4: Constraining a column. a) Top-down view of column with lackey camera to the left and master camera to the right. b) Pillar and destination reference frame $\mathbf{R}_t$, as seen from lackey camera. b) Constrained pillar, $\mathbf{R}_t$, and source reference frame $\mathbf{R}_f$, as seen by the master camera.

Constraints are implemented as an additional transformation within the canonical view volume. We define a spatial constraint matrix $\mathbf{K}$ using the transformation matrices of the two reference frames $\mathbf{R}_f$ and $\mathbf{R}_t$, each of which represents the position, orientation, and scale of the corresponding reference frame in world space. In Cartesian space, the

desired deformation could be accomplished by considering the eye space reference frames, $\mathbf{C}_i^{-1}\mathbf{R}_f$ and $\mathbf{C}_m^{-1}\mathbf{R}_t$. Note that this considers $\mathbf{R}_f$ in the eye space of the lackey camera and $\mathbf{R}_t$ in the eye space of the master camera. Any given point would be first transformed to the local space of this lackey camera eye space $\mathbf{R}_f$. These coordinates would then be mapped to world space using the master camera eye space $\mathbf{R}_t$. The resulting transformation would be $\mathbf{p}'_w = \mathbf{C}_m\mathbf{R}_t(\mathbf{C}_i\mathbf{R}_f)^{-1}\mathbf{p}_w$.

In the special case of having equal projection matrices for the master and lackey cameras, the above transformation would work as desired. However, when the master and lackey cameras have different parameter values, the transformation does not directly map to the space of the canonical view volume, in which points have a homogeneous representation with w typically not equal to one. In particular, the transformation would be attenuated in accordance with the differing focal lengths of the two cameras. As discussed earlier, this is acceptable for the viewport transformation, as the user manipulations to the viewport widget have a relative mapping to the change in deformation, and hence, apparent projection. Constraints, on the other hand, are not designed as relative manipulations, but instead as direct specifications of projective locations. As such, we must convert the above eye space transformation to one within the canonical view volume that accomplishes the desired world space transformation.

To accomplish this conversion, we introduce the operation *cartesianize*. Cartesianize reconstructs a reference frame given by an affine transformation matrix as the equivalent reference frame within the canonical view volume, after projecting with respect to a given projection matrix. Given such a reference frame $\mathbf{R}$, we use the local origin and basis vectors $\mathbf{o_R} =< 0,0,0 >$, $\mathbf{x_R} =< 1,0,0 >$, $\mathbf{y_R} =< 0,1,0 >$, and $\mathbf{z_R} =< 0,0,1 >$. Each of these components representing $\mathbf{R}$ are transformed by $\mathbf{R}$ and multiplied by the associated projection matrix $\mathbf{P}$, resulting in $\mathbf{x}' = \mathbf{PRx}$, $\mathbf{y}' = \mathbf{PRy}$, $\mathbf{z}' = \mathbf{PRz}$, and $\mathbf{o}' = \mathbf{PRo}$. These components are then cartesianized such that they have a Cartesian space form ($w = 1$), and the reference frame is reconstructed with these components.

Finally, we have the following form for cartesianize:

$$cartestianize(\mathbf{R}) = \begin{bmatrix} \mathbf{x}' & \mathbf{y}' & \mathbf{z}' & \mathbf{o}' \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.18}$$

To apply this operation within the constraint transformation, we cartesianize each eye space reference frame, to bring them into a common canonical view volume. The new transformation is as follows:

$$\mathbf{K} = cartesianize(\mathbf{C}_m\mathbf{R}_t)cartesianize((\mathbf{C}_i\mathbf{R}_f)^{-1}) \tag{3.19}$$

To apply the constraint, the resulting deformation transformation for the lackey camera is now similar to Equation 3.16, but with the constraint matrix appropriately inserted:

$$\mathbf{A}_i = \mathbf{C}_m^{-1}\mathbf{K}\mathbf{C}_i. \tag{3.20}$$

This transformation is dependent on the transformations of the constraint reference frames and camera parameters, and can be precomputed before application to all points. Most often, $\mathbf{K}$ is a per-object constraint defined for all lackey cameras, but it can also be global for all objects or defined on a selective basis for each object and lackey camera.

For complex objects it might be necessary to define multiple constraints. Points on the object are constrained to a linear combination of proximal reference frames. Formally stated, a set of constraints $\mathbf{K}_1, .., \mathbf{K}_n$ are defined using frames $\mathbf{R}_{f,1}, .., \mathbf{R}_{f,n}$ and $\mathbf{R}_{t,1}, .., \mathbf{R}_{t,n}$. The constraint matrix $\mathbf{K}(\mathbf{p}_w)$ for a point $\mathbf{p}_w$ is defined using frames $\mathbf{R}_f(\mathbf{p}_w)$ and $\mathbf{R}_t(\mathbf{p}_w)$. $\mathbf{R}_f(\mathbf{p}_w)$ and $\mathbf{R}_t(\mathbf{p}_w)$ are computed as weighted interpolations of frames $\mathbf{R}_{f,1}, .., \mathbf{R}_{f,n}$ and $\mathbf{R}_{t,1}, .., \mathbf{R}_{t,n}$, respectively. The weight for constraint j is inversely proportional[2] to the Euclidean distance from $\mathbf{p}_w$ to the origin of frame $\mathbf{R}_{f,j}$. We precompute $\mathbf{A}_{pre,i} = \mathbf{C}_i = \mathbf{V}_i\mathbf{P}_i\mathbf{E}_i$ and $\mathbf{A}_{post,i} = \mathbf{C}_m^{-1} = (\mathbf{V}_m\mathbf{P}_m\mathbf{E}_m)^{-1}$ to represent the deformation

[2]To avoid division by zero, the weight for constraint j is $1/(1+d)$, where d is the distance from $\mathbf{p}_w$ to the origin of $\mathbf{R}_{f,j}$.

of a point $\mathbf{p}_w$, combining Equations 3.16 and 3.20 as:

$$\mathbf{p}'_w = \mathbf{p}_w + \sum_i (w_i(\mathbf{p}_w)((\mathbf{A}_{post,i})\mathbf{K}(\mathbf{p}_w)(\mathbf{A}_{pre,i}) - \mathbf{I}))\mathbf{p}_w. \tag{3.21}$$

Figure 3.5 shows the importance of constraints. The removal of scene constraints causes the table on the left to undergo a large vertical translation due to the differing positions of the lackey camera and master camera. The ceiling and back wall cave in to the undeformed portion of the scene for similar reasons. In practice, selective nonlinear projections of complex scenes are easy to mangle without a number of constraints to aid in composing a shot in screen space.

(a) (b)

Figure 3.5: Complex nonlinear projection from *Ryan*. a) With constraints. b) Removal of scene constraints causes wall and ceiling to collapse into scene.

3.2.7 Chained Lackey Cameras

Defining a weight-interpolated virtual camera for each point potentially offers better interpolation of the angular parameters of the camera model than the approach we have taken here [57]. For complex scenes as seen in Figure 3.15, the per-control point computation of a virtual camera transformation can be expensive, significantly detracting from the system's interactive capabilities. In practice, we find that blending projected points

provides good visual results, and the matrix precomputations in Equation 3.21 allow for efficient deformation. Interpolation could otherwise be computed as with a more robust transformation interpolation scheme using the exponential map as described by Alexa [4]. An alternative solution is the creation of a chain of in-between lackey cameras that define the interpolation path from the master camera transformation to a target lackey camera transformation. Chained lackey cameras also provide the user with greater control over the illumination blending described in the next section.

3.3 Rendering the Projection

The previous section presented techniques for deforming geometry to appear as a nonlinear projection from the master camera's view. To correctly render a nonlinear projection of a scene, all aspects of the display pipeline must be addressed. This includes not only geometric projection, but also illumination, culling and clipping, and shadow calculation. As a basis for design, these portions of the display pipeline should maintain image coherence over time without introducing visual distractions. View-dependent illumination calculations should appropriately incorporate the different view directions, and view-independent illumination, shadows, and other shading effects should be invariant to projection.

3.3.1 Multiple View Illumination Models

Many global and local illumination calculations are view dependent. A nonlinear projection constructed from multiple views will affect the results of such illumination, as the view vector is now dependent on the local projection. As the perspective model is based on a linear combination of linear projections, an analogous illumination model is defined. Regions of a scene that are projected entirely from a single view are illuminated with respect to that particular view. Geometry that is projected using a combination of

views is illuminated by considering all views that define its projection. Two approaches to combining these views are considered:

1. Illumination is as seen from the weighted interpolation of the view vectors that affect the surface point's projection.

2. Do not create a virtual viewpoint. Illumination is instead a weighted linear combination of the illumination seen from each view direction.

In addition, surfaces are illuminated with respect to their undeformed position in space. Deformation is used only as a tool for creating a desired projection, and the spatial relationship of the deformed geometry is meaningless in regard to illumination.

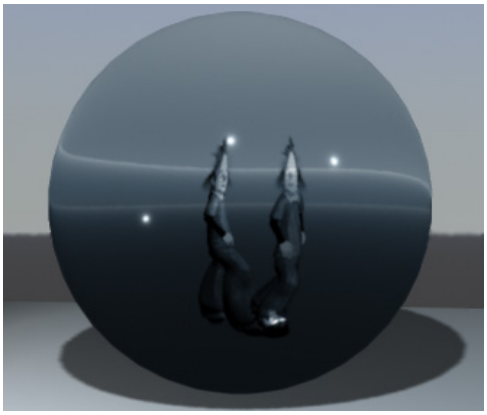
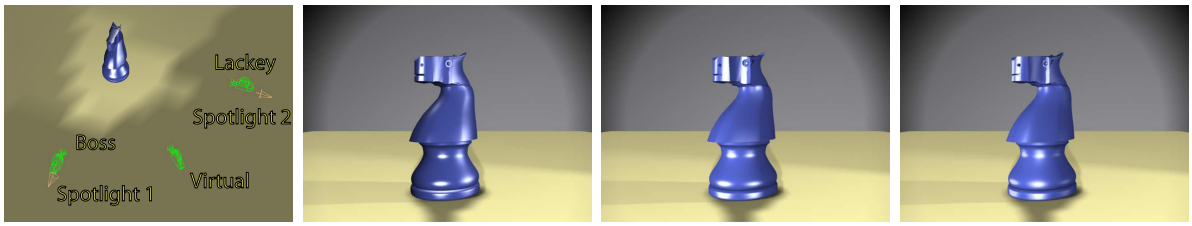


Figure 3.6: Dual reflections using blended illumination.

In practice, the latter approach has proven to be easier to work with. View-sensitive illumination is predictable, as it is a composite of that seen from the multiple views, which are available to the user. Interpolated view vectors can introduce unwanted effects that the user might not be able to predict. In addition, blending illumination calculations allows for the creation of a greater variety of stylistic effects. Figure 3.6 shows an example of this flexibility, where no single viewpoint would be capable of creating the dual views of the character seen reflected in the sphere. In this particular example, the illumination model is isolated from the projection model, and projection is disabled.

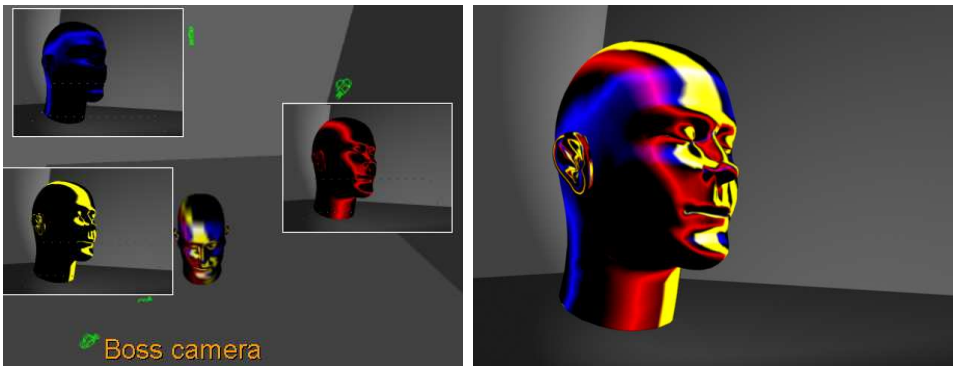


(a) (b) (c) (d)

Figure 3.7: Illuminating a nonlinear projection. a) Scene arrangement. b) Master camera illumination. c) Virtual camera illumination. d) Blended illumination.

As a comparative example, Figure 3.7 shows three variations of illumination for an object viewed with two cameras. The object has been deformed such that when viewed from the master camera, it appears as a linear combination of the master camera's transformation and the lackey camera's transformation. Figure 3.7a depicts the layout of the cameras, original scene geometry, and two spotlights used for illumination. The virtual camera represents an interpolated view direction.

In Figure 3.7b, the geometry is illuminated considering only the master camera's view direction. Note the two highlights: one directly in front of the viewer reflecting spotlight 1 and one halfway to the region directly illuminated by spotlight 2, with no illumination effects due to the presence of the lackey camera. Figures 3.7c and 3.7d demonstrate the two methods by which the illumination from the lackey camera can be incorporated. In Figure 3.7c, an interpolated view direction between the view directions of the master and lackey cameras is used in the illumination calculations, resulting in the two modified highlights. In Figure 3.7d, the object is illuminated with respect to both the master and lackey cameras, and the results are combined, resulting in four attenuated highlights, although two are close enough to appear as a single stretched highlight in this example. This approach can also be applied to stylized shaders, as seen in Figure 3.8.



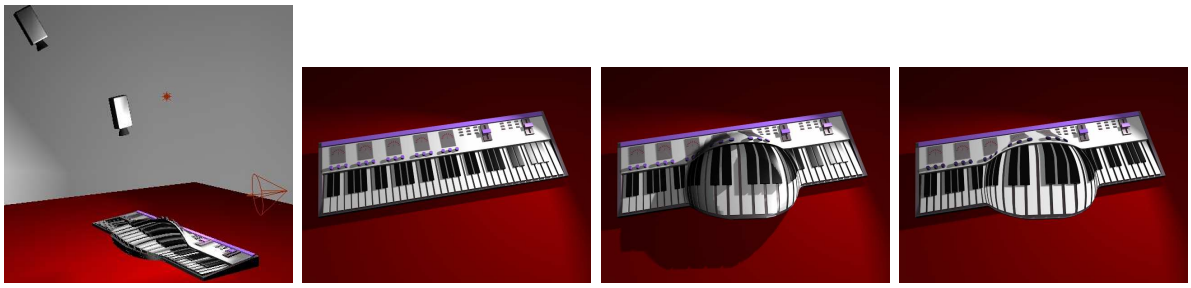
(a) (b)

Figure 3.8: Surface normal based weight computation. a) Scene arrangement. b) Blended illumination.

3.3.2 Shadows

Nonlinear projection conceptually models the visual perception of a 3D scene, imposing a change on the scene. As with illumination, shadows calculated using this deformed geometry are meaningless. In a static image, this results in unwanted and often non-intuitive shadowing. This effect is demonstrated in Figure 3.9c and it is clear that the geometry has been deformed rather than projected. Even subtle perceptual changes created with a nonlinear projection can result in large geometric deformations, which cause this problem to occur surprisingly often. Furthermore, in an animated scene, the shadows cast by deformed geometry will move about the scene, distracting from other important content. To correct this problem, the original geometry is always used for shadow calculations. Pre-computed shadow maps used in multi-pass rendering are generated with the nonlinear projection disabled, and shadow queries reference the corresponding point on the undeformed geometry. Ray traced shadows in off-line renderers should use the undeformed scene for all intersection calculations, with the exception of primary visibility rays. After applying correct shadows, the keyboard appears as in Figure 3.9d. The distracting self-shadowing is no longer present, and only the projection and blended

illumination effects remain.



(a) (b) (c) (d)

Figure 3.9: Shadows in a nonlinear projection. a) Camera setup. b) Master camera view. c) Nonlinear projection without corrected shadows and illumination. d) Correct shadows and illumination.

3.3.3 Geometric Culling

Nonlinear projection has been implemented as the linear perspective projection of deformed scene geometry, allowing us to make use of the underlying graphics pipeline. The linear perspective camera through which the scene is viewed thus handles culling and clipping of objects automatically within the existing graphics architecture. In a dedicated nonlinear projection pipeline, or if our framework were implemented as a procedural displacement shader, care should be taken to consider the final projected position of the geometry in the culling decision process.

In practice, animators construct nonlinear projections while only considering their local effect in space and time. However, the nonlinear projection deformations have a global effect on the scene throughout the timeline, and objects often undergo unwanted deformations to appear when not desired. This happens quite often when deformed geometry is close to a plane through a camera and perpendicular to its view direction. To avoid such artifacts, deformation is attenuated beyond a threshold distance outside

the viewing frustum of the master camera. This threshold distance is under animator control, as the region of space for which deformation is necessary is nontrivial to predict. An animator can also selectively disable this attenuation when using a projection to intentionally bring a distant object into view.

3.4 Implementation

This section describes the implementation of this model as a plugin to the commercial animation system *Maya*. The interface to the system, adapted from the workflow developed by Singh [57], is first presented. The deformation and rendering model as incorporated into *Maya* is then discussed.

3.4.1 User Interface

In the system described above, the user animates a scene with a traditional linear perspective camera–the master camera. Current animation systems such as *Maya* allow users to create, manipulate, and simultaneously view any number of linear perspective cameras. Within our system, the user can at any time label any of these cameras as lackey cameras associated with the master camera. Adding or manipulating lackey cameras appropriately updates the deformation of scene geometry to result in a nonlinear projection as seen from the master camera. An attenuation control on the geometric deformation magnitude allows the user to switch or blend between the nonlinear projection and the master camera's linear perspective view.

When creating and editing nonlinear projections, users typically require an exploratory view of the scene that shows the overall spatial relationship among the scene, master camera, and lackey cameras. All cameras except the master camera view the undeformed scene, since the deformation only has visual meaning from the master camera's point of view. The individual lackey camera views allow the user to visualize the effect of each

camera on the nonlinear projection.

To modify relative depths of different views and apply the viewport transformations, the user interactively manipulates a film box cube coincident in space with each lackey camera. Affine transformations applied to this film box map directly to the viewport transformation matrices used for projection. Translating or scaling the film box along the camera's viewing axis modifies the camera's projection depth relative to other lackey cameras without changing the image space location of projected scene elements.

Users typically create new lackey cameras coincident with the master camera with no viewport transformation as a starting point from which to manipulate the lackey cameras. Lackey camera manipulation can dramatically alter the composition of scene objects in the image. Unless a single nonlinear projection is applied uniformly across the scene, constraints are necessary to tie objects to locations in the final image. Therefore, our system creates a default constraint at the center of each group of objects and each lackey camera with which they are associated. Manipulating the camera thus alters the projection in the master camera view while ensuring that the constraint maps to its location in the master camera's linear perspective projection.

Constraint reference frames are represented as cross-hair objects (*locators* in *Maya*), visible in Figure 3.4. Each pair of these objects defines the constraint's reference frames and can be interactively manipulated as with any scene object. New constraints can be created for a group of objects, and they are typically specified uniquely for each lackey camera. The constraints are largely responsible for allowing nonlinear projections to be easy to control, allowing the overall image composition to remain coherent and predictable. Constraints also provide a direct method for altering the size, position, orientation, or depth of a local region of the scene. The viewport transform does not allow such local changes, as it acts globally for a particular camera. As such, constraints can be defined at any point during the work flow.

3.4.2 Rendering

The nonlinear projection is rendered by interactively deforming the scene geometry and viewing it through the underlying linear perspective of the master camera. The illumination and shading calculations are implemented assuming a shading pipeline in which local surface parameters are provided to the shading system by the renderer. An arbitrary shader calculates the shading and illumination from these parameters and any number of additional parameters that are constant across the surface. The Maya rendering architecture also supports interpolation of user-defined parameters across surfaces, which is necessary for our particular implementation, although the parameters we interpolate could also be calculated directly by the shader. We have constructed two steps in the shading process (analogous to functions within a conventional shading language) that support our illumination model. As shown in Figure 3.10, the renderer provides surface parameters, which we partially replace with the surface point and normal of the undeformed geometry using a reference to the original surface. This takes place for each camera, and the parameter modification also replaces the view vector with that of the corresponding camera. These new values override the renderer-provided parameters to an arbitrary surface shader, thus providing a surface color appropriate for each camera. Another step interpolates among these colors using the weights calculated for projection interpolation, and the resulting color is used to shade the surface. The camera weights are stored as parameters that the renderer interpolates, which are then accessible to the shader. This approach is particularly useful in that any existing surface shader can be used with our model, rather than having to construct specialized versions of each shader.

Ghosting effects as seen in Figure 3.1 are created by duplicating the deformed geometry as nonlinear projection parameters are varied and then rendering the multiple instances of geometry in the scene with varying opacity.

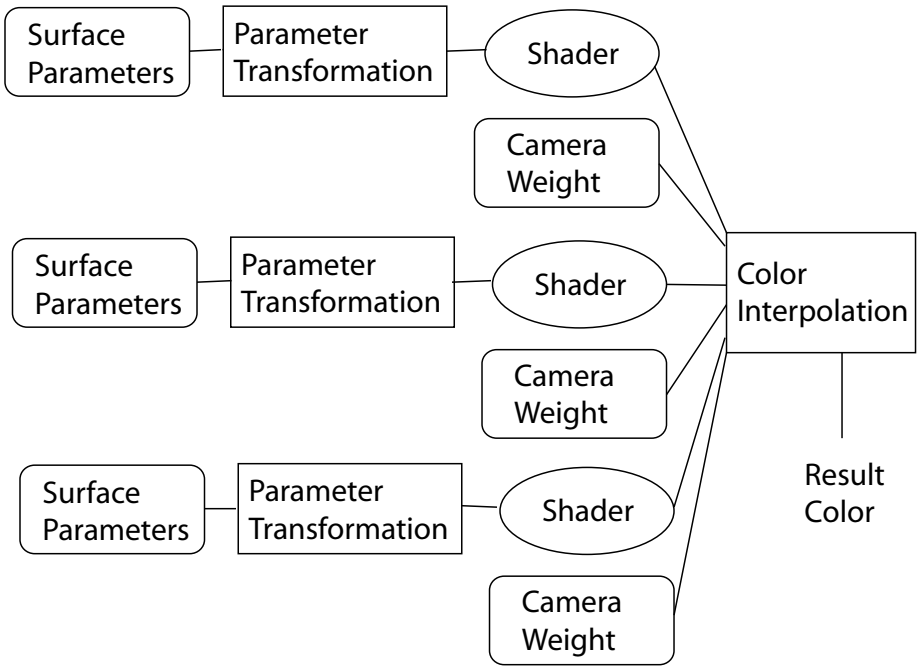
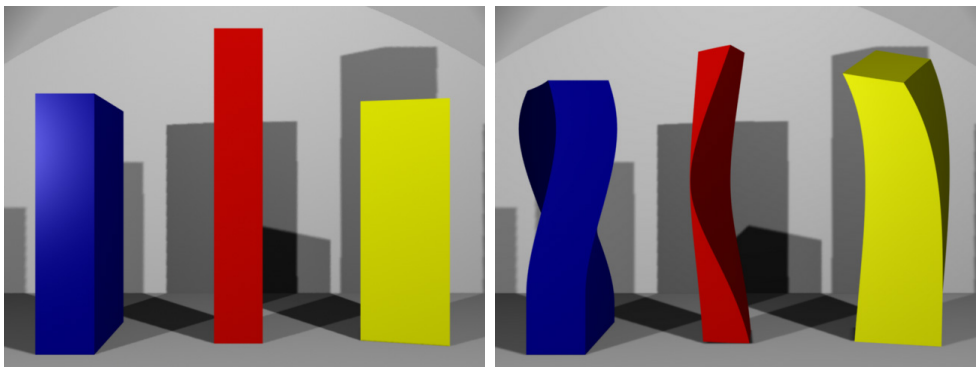


Figure 3.10: Modifications to the shading pipeline. Rounded blocks represent surface parameters provided by the renderer, and rectangles represent new shading functions.

3.5 Results

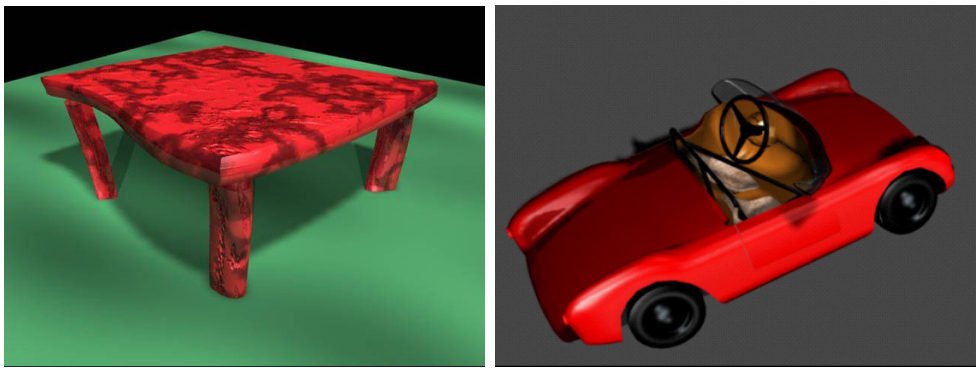
The system described in this chapter allows for the comprehensive creation and rendering of nonlinear projections appropriate for use in a production environment. In Figure 3.11, three blocks have been independently projected to reveal features otherwise not visible. The upper portion of the block on the left is seen from a viewpoint to the left, the center of the middle block is seen from a viewpoint looking in from the right side, and the top of the right block is projected according to an elevated point of view looking down. Figure 3.12 contains two examples of nonlinear projections that depict an object from multiple view directions. Projections such as these can be exceptionally useful in creating renderings of complex objects that reveal features not visible from any single linear perspective, while maintaining a coherent visualization.



(a) (b)

Figure 3.11: Bringing occluded regions into view with nonlinear projection.

The system has been in use in the production of the animated film *Ryan,* demonstrating its artistic and practical usefulness. A rendering of a scene incorporating multiple linear perspectives is shown in Figure 3.13. Figure 3.14 shows another production quality rendering, this time incorporating a global nonlinear projection. The blended illumination model can also be used independently to craft interesting effects, as demonstrated in Figures 3.8 and 3.6. Figures 3.15 and 3.16 show stills from animation tests that employ



(a) (b)

Figure 3.12: Nonlinear projections of objects incorporating multiple views.

nonlinear projections to distort the scene.

To create nonlinear projections, animators work almost exclusively with the master camera for shot composition, but switch among lackey camera views to collect ideas for constructing the nonlinear projection. The ability to gradually apply the existing nonlinear projection to the underlying linear perspective has also proven valuable in both understanding an existing projection during authoring and as a means of subtly introducing nonlinear perspective effects as a shot progresses. Controlling the multiple cameras used by the system can be a complicated task, however, and the next two chapters present higher level techniques for controlling nonlinear projections using this model as a foundation.

Figure 3.13: A multiprojective rendering from *Ryan*.

Figure 3.14: A nonlinear projection rendering from *Ryan*.

Figure 3.15: Animated nonlinear projection of the *Ryan* cafeteria set.

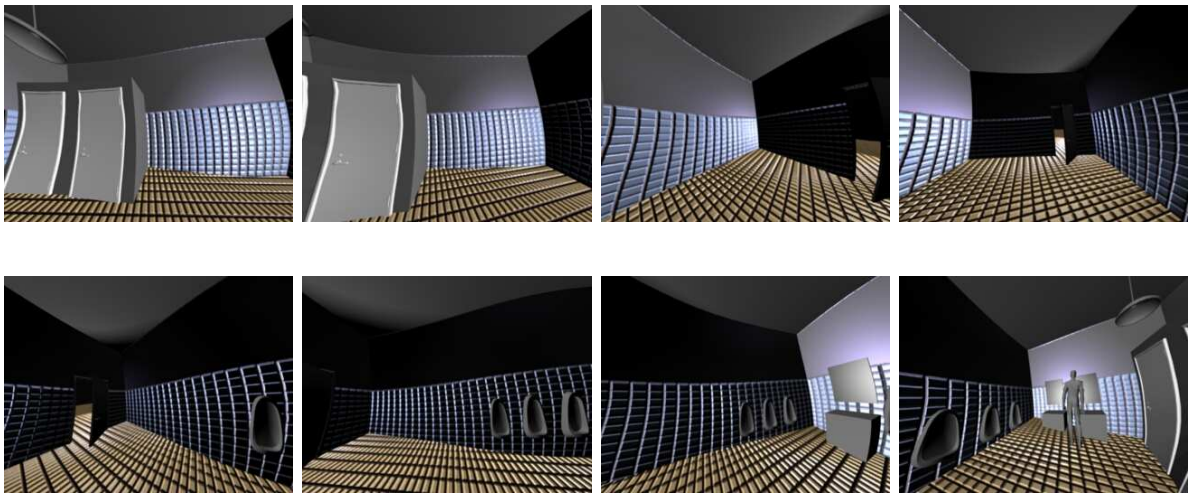


Figure 3.16: Animated nonlinear projection of the *Ryan* bathroom set.

Chapter 4

A Direct Interface for Creating Nonlinear Projections

When working with traditional media, artists typically design scene projections by sketching rough ideas on paper. Motivated by this approach, this chapter presents a system that allows digital artists to lay out, on a 2D canvas, a rough approximation to a desired nonlinear 3D scene projection. A collection of editable feature primitives, along with lines sketched in the 3D scene, are projected through a default linear perspective view to create editable 2D counterparts. The user can then manipulate these editable counterparts to lay out a desired projection of the 3D scene. The original 3D primitives and editable 2D counterparts are mathematically interpreted as constraints on a desired nonlinear scene projection. These constraints are then formulated into a nonlinear optimization problem, which is solved to determine the parameters of a small set of cameras. Each camera is associated with a subset of the constrained features, meeting the associated projection constraints. The projections of these cameras are combined, using the system from the previous chapter, to create a nonlinear projection for the entire scene. The projection of regions of the scene between feature can then be edited by manipulating spatial weight functions associated with each 3D feature. These weight functions are defined as radial

implicit functions, facilitating smooth transitions between linear projections in the resulting image. The parameters of these implicit functions are also under user control, allowing further aesthetic refinement of the nonlinear projection.

This approach to constructing nonlinear 3D scene projections significantly reduces the complexity of working with the many control parameters of multiple cameras. The system runs interactively, allowing users to easily explore nonlinear projections. Users can choose to work with a small set of the most useful feature primitives, or can select from a wider variety as experience is gained. The ability to specify nonlinear projections with sketched 3D curves allows artists to spontaneously and directly manipulate the projection. Sketching metaphors have been used in the past as interfaces to 3D modeling operations [70, 35], but in this context, sketching is used instead to help create a desired projection of existing 3D geometry.

Initially, a large set of cameras is created—one per feature, each camera projecting its associated feature to the location of the edited 2D counterpart. These cameras are refined to a smaller set that approximately meets the user's constraints. This is done to both reduce the computational expense of projection and to reduce the set of weight functions the user can then edit. Nearby constraints in image space are grouped, and if a single camera can meet both constraints, the other is discarded. This process is repeated until removing an additional camera causes an error metric to exceed a threshold value. Each camera's parameters are determined by optimizing with respect to an objective function, which is constructed to meet the projection constraints while penalizing parameter space distance of the cameras from the default view camera. Each constraint has an associated penalty function that penalizes projective deviation of the 3D feature primitives from the edited counterparts. While this approach is general, many common nonlinear projections such as panoramas and detail-in-context fish-eye views are easier to create using feature primitives designed specifically to manipulate their visual qualities.

4.1 Working with Feature Primitives

To directly specify a nonlinear projection, the user begins with a default exploratory view. Feature primitives are created in the 3D scene to lay out the desired projection. The basic set of feature primitives includes object bounding boxes, points in 3D space, and lines that can be sketched onto objects in the 3D scene. These primitives provide a powerful editing set, as they allow the user to control the layout of visually important objects, points, and edges. In Figure 4.1a, a number of these primitives are shown in red. The set is in no way exhaustive, and Section 4.3.2 describes two additional primitives designed for control of common nonlinear projections.

In addition to the default view, an *editing canvas* is provided that can display either the projected 3D feature primitives, their 2D editable counterparts, a rendering of the 3D scene, or the scene with the features overlaid. The projection of the 3D primitives and the rendering are initially projected using the default view camera. The user edits the feature counterparts within the editing canvas to lay out the desired nonlinear projection. This editing canvas is shown in Figure 4.1b, with the projected feature primitives in red and the edited counterparts in green. Finally, the resulting nonlinear projection can also be displayed on the editing canvas instead of the linear projection of the default view camera. Figure 4.1c is an example in which the nonlinear scene projection and edited primitives are displayed. Figure 4.1d shows the final projection without the feature primitive counterparts.

4.2 Defining the Projection

To create a nonlinear projection, the 3D feature primitives and their edited 2D counterparts are used to create a set of cameras. These cameras locally define linear perspective views that project the 3D feature primitives onto their edited counterparts. Achieving this projection is formulated as a nonlinear optimization problem. An objective function

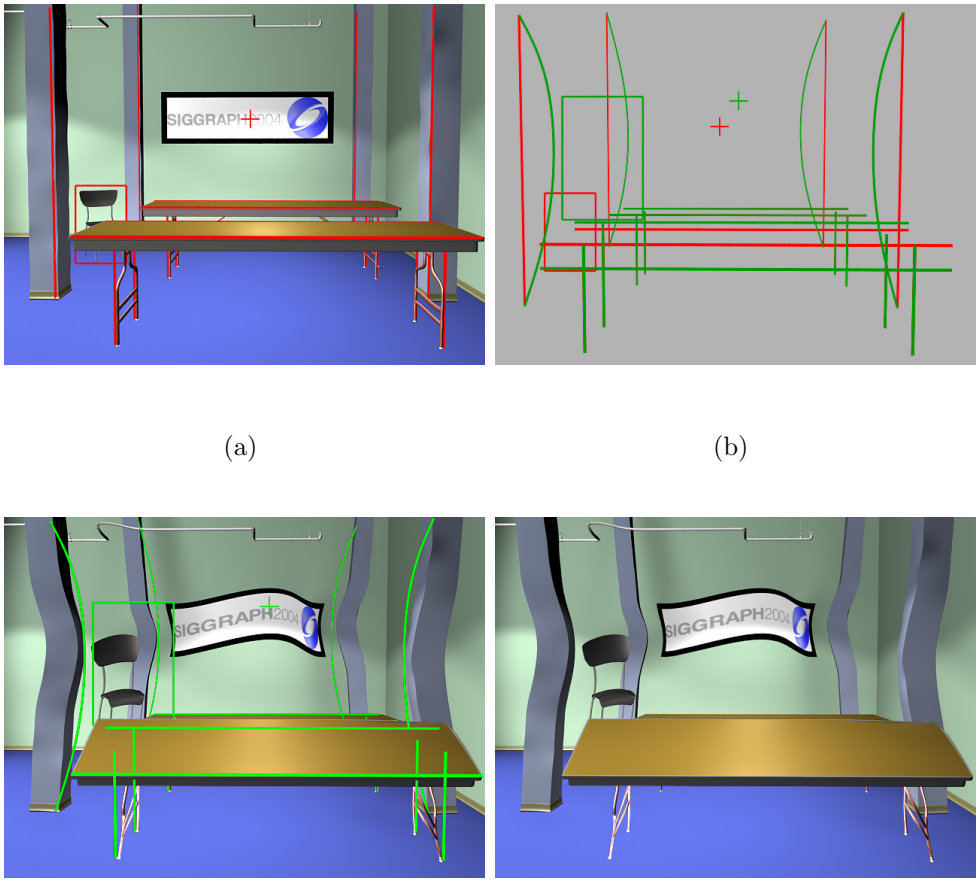


Figure 4.1: Workflow of the direct manipulation interface for specifying nonlinear projections. a) The original scene and a set of 3D feature primitives. b) 2D counterparts of the feature primitives edited to lay out a desired nonlinear projection. c) The generated nonlinear projection with edited feature primitive counterparts. d) The final nonlinear projection. The central regions of the columns are projected to meet the edited curve constraints, although the approximation of the curve constraint as a set of line constraints does not smoothly blend linear perspective views.

is created that penalizes the screen space distance between the projected 3D features and edited counterparts, as well as the parametric distance of the additional cameras from the default view camera. A downhill simplex solver is used in conjunction with a heuristic camera reduction algorithm to create the final set of cameras, along with their associations with the feature primitives. The cameras can then define a nonlinear projection using the system described in Chapter 3, with the cameras' associated spatial weight functions defined relative to the 3D feature primitives.

4.2.1 Constrained Projection Formulation

An objective function is constructed to penalize screen space deviation from both the desired projection, as indicated by the edited feature counterparts, and the default view camera's parameter values. The user should also be able to control the relative importance of the constraints, as some constraints might only be loose indicators of the desired visual layout. In addition, the user should be able to specify how likely each parameter of the additional cameras is to change in relation to the corresponding value associated with the default view. This allows the user to control qualitatively how the projection varies. This can be important, as a local change in focal length has a strikingly different visual quality than change in position and orientation, although either approach might achieve a desired projection.

The algorithm for constructing a projection has two components: manipulating a single camera to best meet an associated set of constraints and refining a large set of cameras to produce a small set that meets the constraints well. At any point during solution, each camera is associated with a group of feature constraints whose projections it defines. A single-camera objective function can be written as follows:

$$E_i = \sum_j w_{i,j} ||p_{i,j} - p_{dv,j}|| + \sum_k w_k E_{i,k} \tag{4.1}$$

In this function, i is a labeling of the cameras, and dv is the label of the default view

camera. j is an index over the free parameters of the camera model; $p_{i,j}$ thus denotes the j^{th} parameter of camera i. k is a labeling of the constrained feature primitives, and $E_{i,k}$ is a penalty function defined appropriately for each type of feature primitive. For any given type of primitive, $E_{i,k}$ is constructed to penalize the screen space distance from the projection of the 3D primitive to the edited 2D counterpart. These are described in more detail in Section 4.3. If a feature primitive is *not* associated with a given camera, $E_{i,k}$ is defined to be zero.

For maximum flexibility, an interface could be constructed whereby each camera has independent penalization of the parametric distance from the default view camera. In this case, $w_{i,j}$ would be tied to a per-camera user control. In our example interface, these weight functions are equal for a given parameter across all cameras, with a single w_j for each parameter. This allows an arbitrary number of cameras to be created as needed, and allows the presence of the multiple cameras to be hidden from the user. The free parameters can be any subset of the parameters used to construct the viewing, projection, and viewport transformations. The system described here incorporates the perspective projection matrix described in Chapter 2, with focal length being the only free parameter. The camera position, direction, and tilt are the free parameters associated with the viewing transformation, as this is a common control space for manipulating computer graphics cameras. The tendency of a camera parameter to change increases as its corresponding weight function decreases, allowing the user to define a relative tendency for deviation among the camera parameters. The user can similarly manipulate the coefficients of the feature penalty functions to assign a relative importance relationship among the features; those with higher weight values are more likely to be met.

4.2.2 Solving for a Multi-Camera Projection

Initially, each feature primitive has a single camera that is associated with only that feature privative and meets its constraint exactly. The parameters of any one of these

cameras are determined by optimizing the objective function defined above with respect to the parameters, considering only the single associated feature primitive. Given this one-to-one mapping of cameras to feature primitives, we wish to reduce the camera set to a minimal set that meets all the constraints to within a threshold value of the objective function, as summed across the cameras. The camera reduction is achieved by repeatedly grouping the constraints associated with two cameras and fitting a single camera. If the resulting value of the overall objective function is less than the threshold, the grouping is accepted. Otherwise, the grouping is discarded and the original two cameras remain in the set. To avoid considering all possible pair-wise groupings among cameras at each step, we take a heuristic approach to choosing associations.

Camera groups are considered by using the heuristic that features near each other in the editing canvas will have similar edits. These groupings are thus more likely to result in small increases to the overall objective function when a single camera is fit to the associated constraints. The entire set of features, K, defines a set of potential pairs $\kappa_{ij} \in K \times K$. These potential pairs are sorted into non-decreasing order, using the metric of total screen space distance among nearest feature primitives. Denote this sorted sequence as the *unresolved* sequence F. By convention, any feature in an entry of F can still be regrouped. We then repeatedly select an arbitrary feature primitive f from the first element of F and attempt to group it with all other unresolved features. This is done by selecting pairs from F that contain f, in order. Each potential grouping has a single camera fit to it, using optimization of the single camera objective function across all feature pairs in the group. If the overall objective function value is above the threshold value, the grouping is not accepted. If it is below, the grouping is accepted. Additionally, all additional pairs in the unresolved sequence F that contain f are removed, avoiding the possibility of f being grouped again. This approach maintains the one-to-many relationship from cameras to feature primitives, while taking a greedy approach to maintaining approximate local linear perspective among the features.

To solve the nonlinear optimization problem, we use the downhill simplex method [47]. While not the most efficient approach, it works well when the penalty coefficients can have large variance in magnitude and the gradient of the objective function is not available. A simplex is defined in parameter space from the initial parameter values, and a set of refinement operations are used to contract the simplex until the approximate error falls below a given threshold. For the initial parameter values, we use those of the default view camera.

4.3 Editable Feature Primitives

Feature primitives can serve one of two functions: specifying local constraints on the projection of specific 3D scene locations or direct specification of the global visual qualities of the projection. The former case uses features associated with 3D space and the projections of these features to construct the penalty functions. The latter case uses manipulations of 2D primitives at a given depth, but this can be expressed in terms of the former by associating the unedited location of the 2D features with a corresponding 3D primitive. All 3D primitives are then used to define the weight functions, as described in the next section.

4.3.1 Composition and Projection Constraints

Constrained feature primitives expressed on 3D scene geometry reflect the user's desire about what should project to where to achieve a desired composition. We use three primary types of constraint to achieve this:

- **Points**: The simplest form of feature constraint, this expresses the desire that a location in 3D space should project to a particular location in 2D space. The associated penalty function is the screen space distance between the projected 3D

location $\mathbf{Pp}_{feature}$ and the location of the edited 2D point $\mathbf{p}_{edit}$:

$$E_{point} = ||\mathbf{Pp}_{feature} - \mathbf{p}_{edit}||$$

M represents the viewing, projection, and viewport transformations of the camera associated with this primitive (this is true for all types of constraints). The initial location of $\mathbf{p}_{edit}$ is $\mathbf{Pp}_{feature}$

- **Lines**: This constraint indicates that a linear 3D region should map to a linear 2D region. Depending on how the projected line differs from its edited counterpart, this feature can be used to locally change the view direction, location, or focal length. The penalty function is a combination of point constraints at the endpoints $\mathbf{p}$ and $\mathbf{q}$:

$$E_{line} = ||\mathbf{Pp}_{feature} - \mathbf{p}_{edit}|| + ||\mathbf{Pq}_{feature} - \mathbf{q}_{edit}||$$

- **Bounding Boxes**: This primitive allows the user to express a desired position and size of an object in the scene. The penalty function is expressed in terms of the bounding box center $\mathbf{p}$, projected width w, and projected height h:

$$E_{bb} = ||\mathbf{Pp}_{feature} - \mathbf{p}_{edit}|| + |w_{feature} - w_{edit}| + |h_{feature} - h_{edit}|$$

Each of these feature primitives is demonstrated in Figure 4.1. The tables are constrained by line primitives, the center of the sign is constrained by a point primitive, and the chair is constrained by a bounding box primitive.

4.3.2 Feature Primitives for Common Nonlinear Projections

We consider two common nonlinear projections: panoramas and fish eye views. Panoramas are characterized by a continuous change in perspective along one dimension. Fish-eye views are characterized by a change in view direction with respect to the distance from the center of the image.

- **Panoramic**: To specify this projection, a line is is edited to have a curved shape. The curve is then discretized into a number of linear segments, each of which is expressed as a single line constraint. Typically, a number of cameras are generated that influence local regions along the line constraint, and the cameras' regions of influence will blend together if the line constraints meet at the endpoints. This type of constraint is demonstrated in the projection of the columns in Figure 4.1.

- **Fish eye**: This projection is specified using boxes, which are initially concentric. They are defined relative to some point in space, and the boxes operate as bounding box constraints. In this case, two cameras are used; the camera associated with the inner bounding box can only pan or translate in depth with respect to the camera associated with the outer box. A custom weight function is defined such that there is a gradual change of weight from one camera to the other in the region between the two boxes relative to image space. Figure 4.2 illustrates the use of this constraint to create a projection where the center of the image is seen from a distant view.

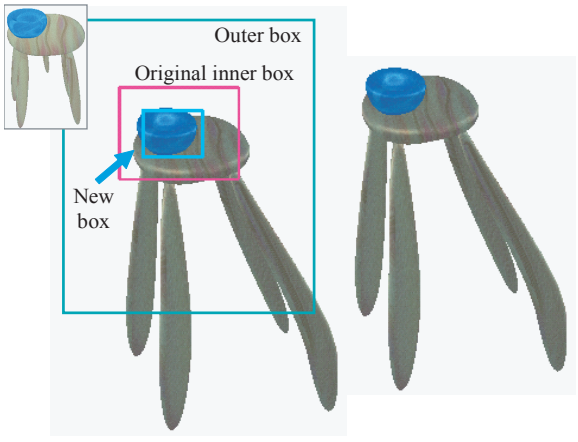


Figure 4.2: Feature primitive for designing fish eye projections.

4.4 Primitive-Based Weight Functions

The above sections address the placement of cameras that define local linear projections of scene geometry. While this defines the projection of the features themselves, the projection of the remainder of the scene still needs to be defined. To maintain local characteristics of linear perspective, we desire regions of space near a feature primitive to project similarly to the primitive itself. As distance from a primitive increases, the projection will vary to approach that of the default view, and this should be a smooth transition.

To address this problem, we define spatial weight functions in terms of the features as offset implicit functions. These functions have a value of one within a specified distance from the primitives, and they fall off smoothly beyond this distance. For each camera, we define a weight function for each feature primitive associated with it. The overall camera weight function is then a blended sum of the weight functions associated with the camera's feature primitives. We use two offset values, r_{in} and r_{out}, to denote the inner and outer bounds of the falloff region with respect to the distance from a primitive. As with the radial falloff function used in Chapter 3, the function should have derivatives of zero at the boundaries of the falloff region. The resulting function, where R denotes the distance in space from a feature primitive, is:

$$f(R) = \begin{cases} 1 & \text{if } R < r_{in} \\ 2R^3 - 3R^2 + 1 & \text{if } r_{in} \leq R \leq r_{out} \\ 0 & \text{otherwise} \end{cases} \tag{4.2}$$

Each camera now has an associated weight function in space that varies from zero to one with smooth transitions. As in Chapter 3, we add the weights of the various cameras to define the projection as a linear combination of the projections of the multiple cameras. To enforce that weight functions are a barycentric combination among the various cameras, the default view's weight is set to be dependent on the sum of the other

cameras' weights as follows:

$$w_{dv}(\mathbf{p}) = (1 - \sum_i w_i(\mathbf{p})) \tag{4.3}$$

The parameters r_{in} and r_{out} are under user control to allow for the adjustment of the weight functions as necessary to create a desired transition in projection among features. Figure 4.3 shows the 0.8-valued isocontours of the weight functions for the various cameras (each camera having a different color). An example edit is shown in Figure 4.4. First the point constraint has a small region of influence, and there are noticeable nonlinearities in the projection of the sign. By increasing the region of influence (Figure 4.4b), the region containing the sign has a linear projection due to the camera associated with the point constraint having a larger region of full weight.

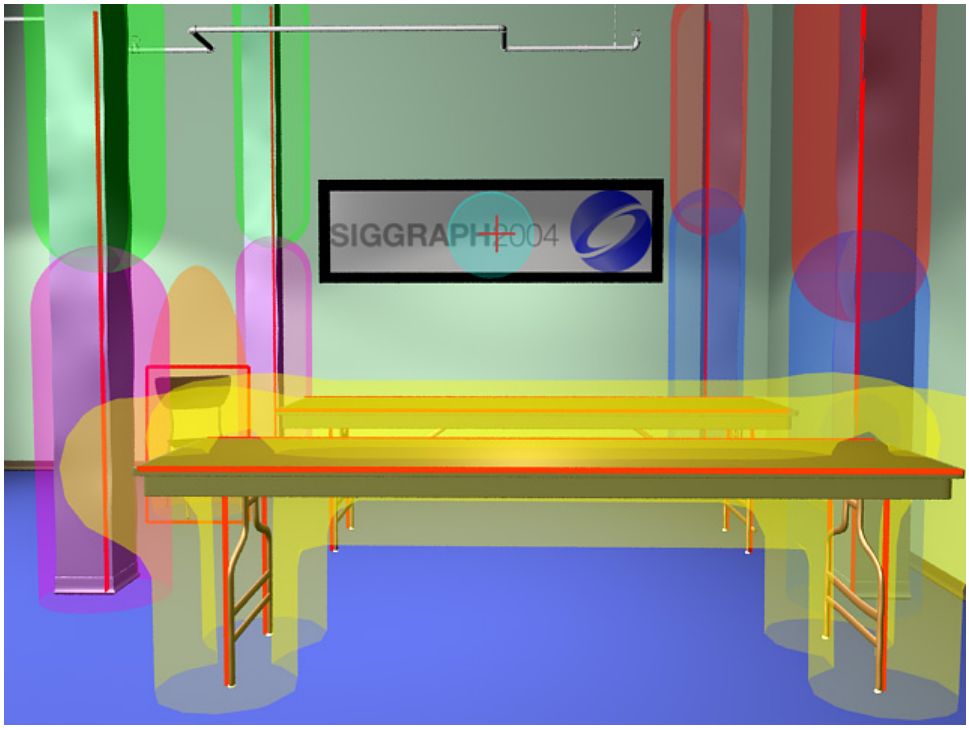
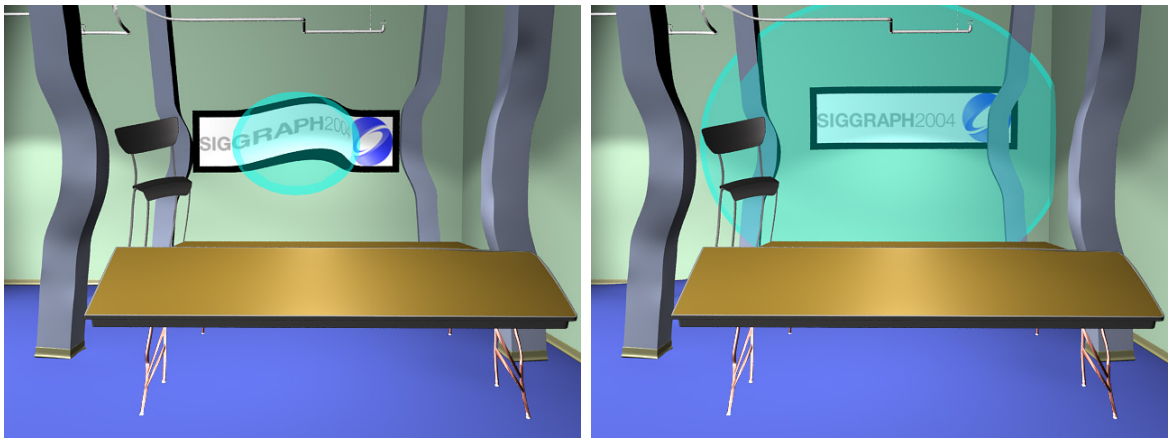


Figure 4.3: Weight function isosurfaces.



(a) (b)

Figure 4.4: Editing a weight function primitive. a) The point feature centered on the picture has a small region of influence. b) Increasing the point feature's region of influence.

4.5 Discussion

This chapter has presented techniques for directly specifying constraints on and visual qualities of a desired projection. While the approach is suitable for creating a static image, it is inherently limited for creating animated projections. Projections could be key-framed in time, but, in general, there will be no correspondence among the number of cameras at different key projections from which to interpolate. Even if a fixed set of cameras were to be enforced, it is likely that the camera-feature relationships would change among the key projections. The next chapter presents procedural approaches to creating projections that better handle animated projections, but they does not allow for the direct specification of projective qualities.

The approach to defining panoramic projections with a discrete set of line constraints is also less than ideal. While local regions will project as desired, the resulting projections do not have the smooth transition implied by the edited curve (Figure 4.1c). A better approach would be to locally specify cameras along the curve and use a high order

interpolation scheme to transition among the camera weights with respect to the unedited curve.

Chapter 5

Procedural Models of Projection

The previous chapter described a technique for high-level control of nonlinear projections based on the direct specification of local projective qualities. While it works well for static nonlinear projections, it is difficult to adapt to projections that change with time. To address this limitation, this chapter explores approaches to procedurally modeling nonlinear projections that are suitable for animation. The procedural projection models described here are designed to allow animators skilled with manipulating a single camera to easily adapt their skills to the multiple-camera paradigm. To this end, the presence of multiple cameras, constraints, and weight functions are encapsulated within the individual projection models. The animator works with a small set of intuitive parameters and only a single master camera. Additional cameras have their transformation and viewing parameters defined relative to those of the master camera, allowing a projection to be manipulated independently of animated shot composition. Each projection model has been designed to explore a particular potential application. Within any procedural projection model, a lower level camera intuitively represents a projective refinement relative to the master camera (or any intermediate ancestors). Underneath each procedural model, the nonlinear projection system from Chapter 3 is used to accomplish the projection as scene deformation.

Straightforward procedural projection models control only the multiple cameras and the high-level parameters. Incorporating scene knowledge into the projection model could potentially offer more interesting applications, as projections can be developed that adapt to scene content. As an example, disjoint visualizations of an action using multiple angles and replays are integral to presentations of team sport activities. These techniques are sufficient for a passive audience, but they are less appropriate for users of interactive application, where real-time feedback is ideal. Automated nonlinear projections might offer an approach to addressing this problem. In addition, the multiple views that are possible within a single nonlinear projection can integrate different story elements and imply or emphasize relationships in scenes with complex arrangements. Projections that adapt to emphasize objects of interest could be powerful creative tools for manipulating how a viewer understands a scene.

This approach to high level control thus simplifies the use of multiple-view projections. The underlying control techniques of a particular projection model are encapsulated within few parameters, and the predictability of well-constructed models offers a convenient interface for keyframe animation. The complexities and technical difficulties associated with controlling complicated arrangements of cameras, constraints, and projection weights are isolated from the user, allowing him to focus instead on creative effort.

5.1 Hierarchical Projection Control

As one example of procedural control, a hierarchical camera structure has been developed. While this hierarchical camera framework has been designed as a general tool to support many specific types of projection, this thesis only presents an exploratory initial model, the camera mosaic, to demonstrate its functionality. Cameras in the hierarchy are controlled relative to their parent camera. Camera parameters, by default, are inherited

from the parent, but they can be individually altered, relative to the parent's parameter values. Every camera also has an associated region of space over which it influences the overall nonlinear projection. This region of influence is defined as a strict subset of the parent's region of influence. The nonlinear projection resulting from this hierarchy can be interpreted as a successive local refinement of linear perspective as one descends the camera hierarchy. The top level is equivalent to the linear perspective view of the master camera. Each level down the tree represents a partition of the parent cameras region of influence among the child cameras. This partition is strictly hierarchical; that is, no camera influences regions of a scene not influenced by ancestors in the hierarchy.

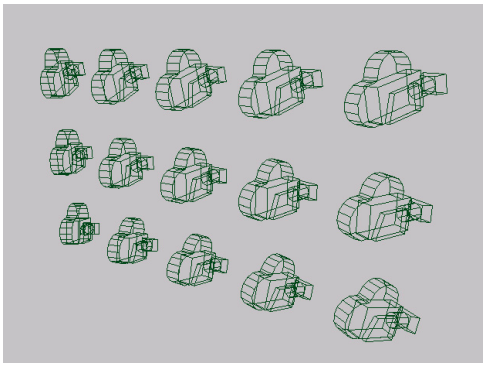


Figure 5.1: Example camera hierarchy. The master camera is coincident in space with the camera at the center. The projection of the scene is defined in terms of a partition of the master camera's view frustum among the child cameras.

A single level camera hierarchy is shown in Figure 5.1. In this example, the center camera is coincident in space with the master camera, and each child camera is translated and rotated relative to the master camera. This results in a camera array that refines projection in a grid pattern, approximating a form of mosaic vision. This projection model is further described in Section 5.1.3.

5.1.1 Model of Projection

The relationship between parent and child cameras at each level is analogous to the relationship between a master camera and lackey cameras in the projection model from Chapter 3. This extends the deformation of a point due to any given camera to a recursive form. Let the deformed point for a leaf camera in the hierarchy be $deformed(\mathbf{p}_w) = \mathbf{p}_w$. This is equivalent to no nonlinear projection deformation, as the view is a linear perspective view relative to the leaf camera. For any camera with children, the unweighted, world-space deformation that causes the parent to see point as the child camera sees it is $\mathbf{A}_i = \mathbf{C}_{par}^{-1}\mathbf{C}_i$. This is mathematically equivalent to the master-lackey relationship. Introducing the weight functions $w_i(\mathbf{p}_w)$, defined for each child camera, the deformation of a world space point (previously Equation 3.16) is then as follows:

$$deformed(\mathbf{p}_w) = \mathbf{p}_w + \sum_i w_i(\mathbf{p}_w) * (\mathbf{A}_i deformed_i(\mathbf{p}_w) - \mathbf{p}_w). \tag{5.1}$$

The weight contribution of any leaf camera to the overall projection is modulated by the weight functions of its ancestors (see Equation 5.1). This simplifies the management of the camera influence weights within any projection model. The projection of any subtree in the hierarchy is thus a nonlinear projective refinement of the local linear perspective of the parent camera. Constraints are similarly defined from a hierarchical standpoint, as they define the composition of a child camera's view of the scene within its parent's view.

5.1.2 Implementation

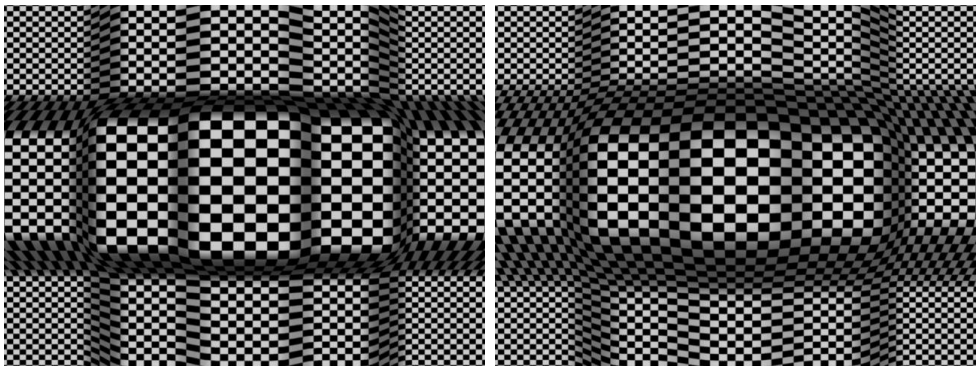
The above framework and the mosaic example presented in the next subsection have been implemented as plugins to the *Maya* animation system. The animated master camera, procedural parameters, and 3D scene information are available to the projection model. The model's associated control algorithm (an example of which is the perspective mosaic

described in Section 5.1.3) controls the relative parameters within the camera hierarchy, the associated weight functions, the partition a camera's region of influence among its children, and a per-camera set of composition constraints. The transformations of the hierarchy are directly mapped to cameras, and constraints are arranged within a similar transformation hierarchy. The resulting world space parameters of the cameras and constraints are used by the nonlinear projection system from Chapter 3 to deform 3D scene geometry. The result is an animated nonlinear projection seen through the master camera. The design allows animations authored conventionally with a single camera to be transformed into animated nonlinear projections by applying a hierarchy, hidden from the user, who is only aware of a small set of control parameters.

5.1.3 Stylized Projection

The hierarchical camera structure lends itself well to developing stylized projections of 3D scenes. As an example, a projection model for continuous perspective mosaics has been created. This model is inspired by both Hockney's photo-mosaics [33] and Escher's use of continuously changing local linear perspective [19]. Perspective mosaics partition the region of influence of a camera into local regions of linear perspective, laid out in a grid pattern in image space. Blending regions are defined between adjacent views to avoid perspective discontinuities, which would otherwise appear as image edges in a traditional photomosaic. The blend is defined as a monotonic cubic interpolation among adjacent views (see Figure 5.5a). In the example images shown in this section, the blending regions between adjacent views have been emphasized with darkened shading.

In this projection model, the animator has control over the dimensions of the grid of linear perspective views and the proportional width and height of each view relative to the regions of overlap. Each view is projected according to the perspective of a single associated camera. Constraints are defined such that a single line of sight is maintained through the center of the local view. This is necessary for each view to appear in the

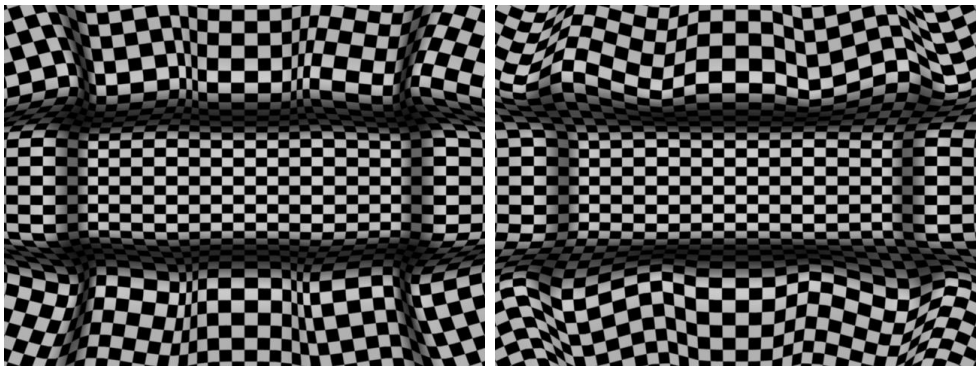


(a) (b)

Figure 5.2: Perspective mosaic. A grid of cameras arranged relative to a master camera locally change scene projection. In this and subsequent examples, a flat plane orthogonal to the view vector is used illustrate model parameters. In this example, camera further from the center are incrementally translated back relative to the master camera, resulting in the plane appearing more distant in the corners. a) Narrow blend regions among adjacent views. b) Wide blend regions among adjacent views.

appropriate image space location.

The animator can manipulate the mosaic projection model using parameters to control horizontal and vertical translation, horizontal and vertical rotation, tilt (rotation about the view vector), dolly (depth translation), and focal length. Each is defined incrementally per camera, and is relative to the position, orientation, and perspective parameters of the master camera. Each parameter's maximum value is associated with the outer cameras in the grid. The incremental values accumulate with grid distance from the master camera. Thus, a camera at the center of the grid has no change relative to the master camera. Cameras associated with the grid corners are the most different from the master camera. In addition to these incremental changes in camera parameter values, control over the width of blend regions is provided as a percentage of the image space grid layout. Figure 5.1 shows the horizontal rotation value accumulated across the



(a) (b)

Figure 5.3: Manipulating the incremental rotation in the mosaic grid projection model. a) Incremental rotation outwards. b) Incremental rotation inwards.

rows of the camera grid to capture a wider field of view than is possible using a single camera without introducing severe perspective distortions.

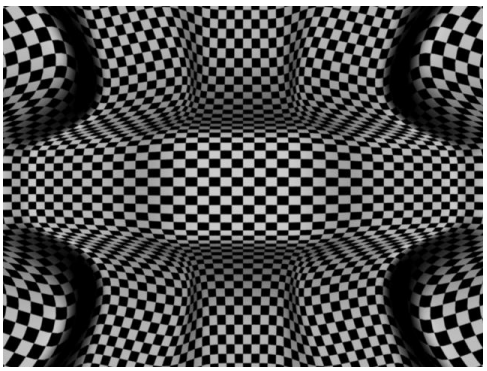
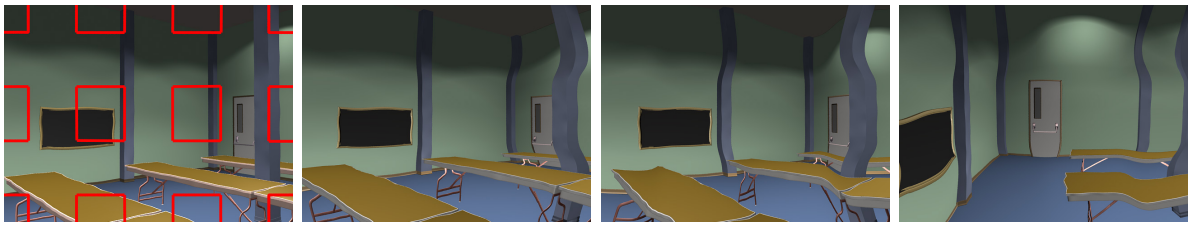


Figure 5.4: Combining mosaic parameters. Each region of the scene is viewed from a different position, orientation, and depth translation, all defined relative to the master camera.

Figure 5.2 demonstrates a 5 x 3 perspective mosaic with a incrementally increasing dolly back in space relative to the master camera. The central camera is coincident with the master camera and the remaining cameras are translated back along the viewing axis based on their grid distance from the central camera. As a result, the checker pattern

in the center of Figures 5.2a,b is larger than the pattern on the fringes of the image, as it is seen from a closer vantage point. Darker shading indicates the regions of blended perspective at boundaries of the weight partition among the cameras. Figure 5.2b has a smoother transition of local perspective than Figure 5.2a, caused by increasing the relative size of the regions of overlap. Figure 5.3 shows the effect of varying the tilt and rotation parameters, and Figure 5.4 illustrates the perspective mosaic effect with a combination of incremental change in translation, rotation, and focal length.



(a) (b) (c) (d)

Figure 5.5: Perspective mosaic applied to a set from the film *Ryan*. a) Mosaic grid with linear perspective view. b) Increasing vertical rotational component. c) Increasing horizontal rotational component. d) Invariance with respect to camera motion. The animator can move a camera, and the projection model will follow. This has the same projection as a-c, but the master camera has been moved.

Perspective mosaics have a small parameter space interface that is designed for creative exploration. Figures 5.5b and 5.5c illustrate different settings of these parameters using a 3D scene. Figure 5.5d, when compared to Figure 5.5c, demonstrates that the projection is independent of the position of the master camera and other 3D scene elements. The procedural projection models in the next section do no have this property. Figure 5.6 is another example using this projection model.

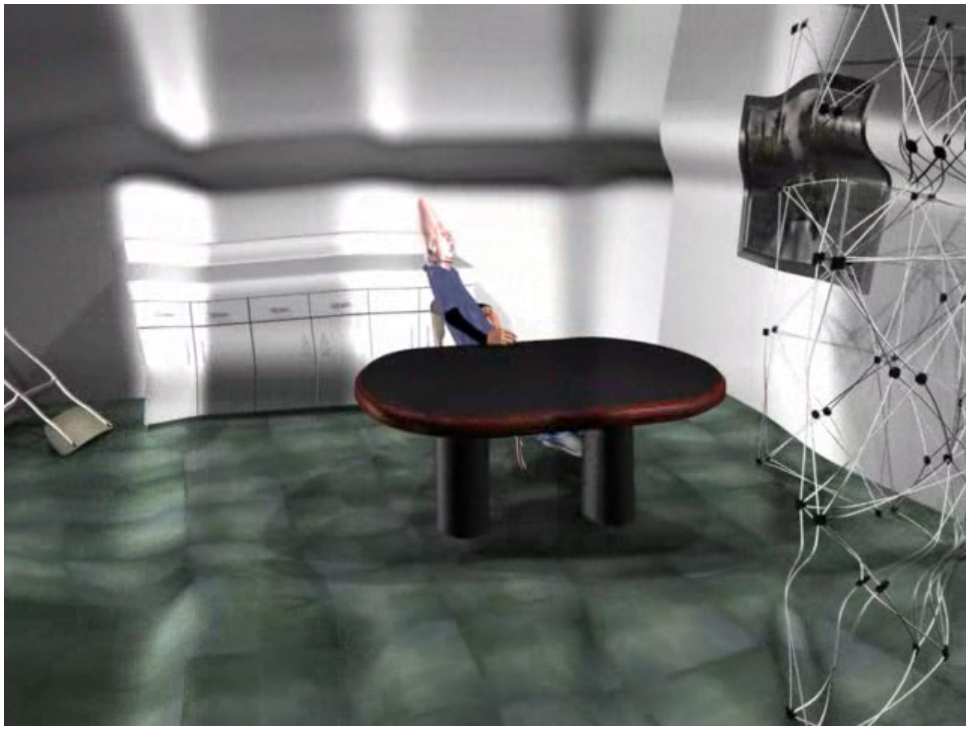


Figure 5.6: Perspective mosaic of an 3D scene. As with previous examples, darkened shading is used to emphasize blending regions between adjacent views.

5.2 Cameras in Motion

Interactive graphics applications often rapidly switch among multiple camera views. Even in traditional editing, camera cuts are common, especially when filming dialogue sequences [37]. Most often, cuts are composed such that the camera never crosses the *line of action*, an imaginary line connecting subjects of interest. This approach maintains a common scene composition between adjacent shots, demonstrated by the two views in Figure 5.7. Two approaches to rapidly switching between adjacent camera views by interpolating projection parameters have been developed. The animated nonlinear projection produces a continuous transition between views, potentially increasing the understandability of transitions by providing intermediate context.

Figure 5.8 illustrates a simple approach to blending between camera projections. Rather than cutting from one camera view to another, the entire scene is nonlinearly

projected as an interpolation between the two camera views over a small number of frames (about half a second). This technique easily extends to blending among more than two camera views or key values of an animated camera. The resulting effect uses stylized projection to indicate camera motion, similar to how motion blur in used to illustrate object movement. The appearance of the scene is similar to a nonlinear squash or stretch proportional to the relative speed and direction of a moving camera. Each of the multiple discrete views (in this example, two) has a region of influence in time, with a smooth blending between views. This results in animated camera projection that appears similar to a panning and turning camera *with no actual camera motion.* When more than two cameras are used, camera weights are interpolated through time with an approximating polynomial.

We can similarly construct nonlinear projections for animated cameras whose weight values progressively increase across space, rather than uniformly across the entire scene. Figure 5.9 shows a transition similar to that above, but the change of view progresses in depth and increases in magnitude for the second camera as time increases. An image-space weight function is used to increase the contribution of the second camera's projection relative to a central point. The resulting transition brings objects seen from the second camera into view as time progresses. The two cameras are positioned rotationally outward from a central master camera used to control the camera rig such that all three share a common center of interest and distance to the center of interest.

5.3 The Scene in Motion

While nonlinear projections designed for stylistic purposes can be used for many creative applications, incorporating knowledge of scene information can allow them to adapt to changing 3D content. The motion of relevant scene objects, in particular, can be emphasized or exaggerated for creative or illustrative effect. In particular, local projection

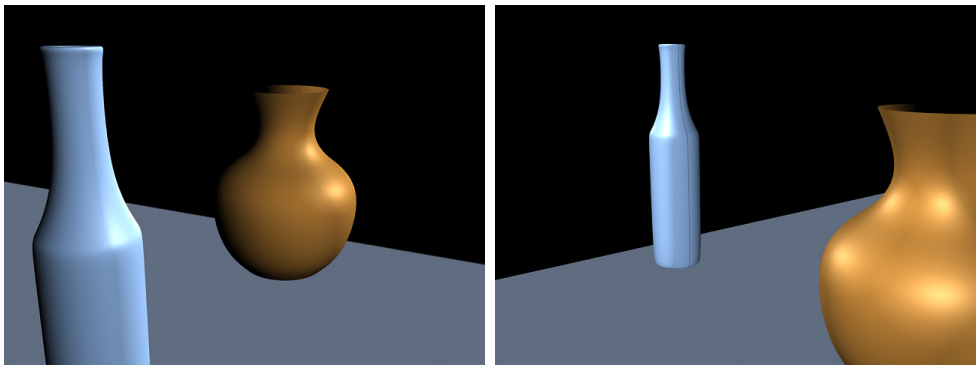


Figure 5.7: Two shots from an example cut.

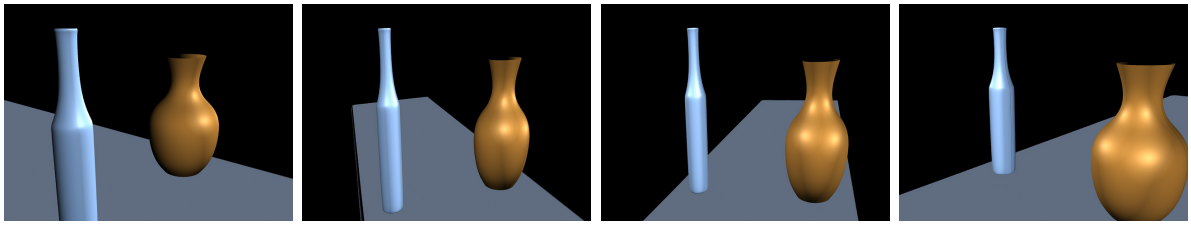


Figure 5.8: Intermediate frames of an interpolated projection cut. Over time, the projections of the two cameras are gradually changed such that neither camera moves, yet a stylized appearance of motion remains.

can be manipulated for emphasis by varying viewing position, direction, and focal length.

Dynamically manipulating the perspective of moving objects can lead to views that present the audience with potentially better perception and understanding of a given motion. In this model for adapting projection to moving objects, the master camera is equivalent to the traditionally animated camera. Moving objects whose speed relative to the scene exceeds a threshold value are assigned an associated local camera. The positions and view direction of these local cameras are manipulated such that the motion is seen from a alternate vantage point, defined using two heuristically chosen manipulations. The first approach rotates the local camera to a view orthogonal to the motion vector. The local camera viewpoint is rotated out from the master camera such that a common center of interest is maintained at the center of the object. The other manipulation applied to the local camera is an increase in focal length, proportional to the object's

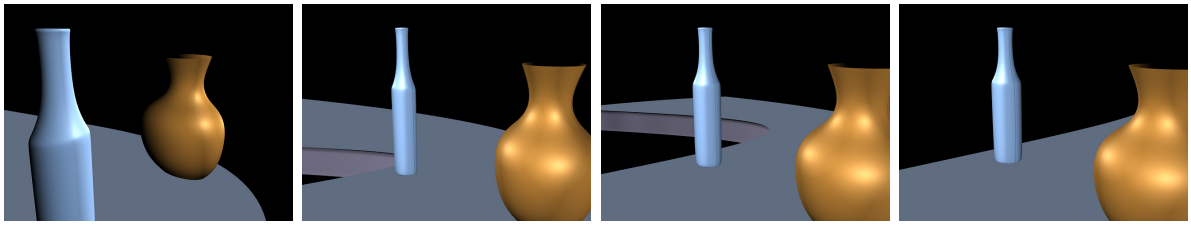


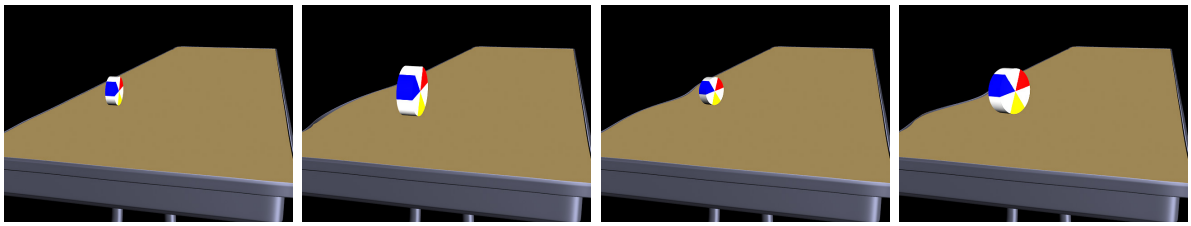
Figure 5.9: Intermediate frames of image-space weighted projection cut. In addition to the gradual change of weight over time as seen in the previous example, camera weights gradually change across the image over time.

speed. This can be used as either an alternative or complementary method of projective emphasis. When the object is moving with low velocity, the child camera weight values are near zero, inducing a subtle effect. The local camera's weight scales up to full local influence when the object reaches as user-specified maximum velocity. Weight values fall off in space radially about the center of the moving object. A positional constraint is used to keep the region of manipulated projection centered relative to its undeformed location.

As an example, Figure 5.10 shows a cylinder rolling along a table, heading toward the master camera. Figure 5.10a illustrates the space near the object as seen from the master camera. In Figure 5.10b, a local camera is created to track the motion, and it has an increased focal length to zoom into the region of space near the cylinder. Figure 5.10c illustrates the rotation of the local camera from the master camera's position relative to the center of the object. The last frame (Figure 5.10d) incorporates a combination of these two techniques to both emphasize the motion and present it from a vantage point that presents more motion content within the final image.

5.4 Discussion

This chapter has presented techniques for procedurally controlling nonlinear projections. The procedural control allows complex nonlinear projection models to be encapsulated



(a) (b) (c) (d)

Figure 5.10: Manipulating perspective to emphasize motion. a) Linear perspective view of a rolling cylinder. b) Increase of focal length to increase the apparent size of the region of space surrounding the object. c) View rotation to increase motion apparent in the image. d) Combining both effects.

within a small set of intuitive parameters. This, in conjunction with the ability to control a projection model my manipulating a single master camera alone, allows animators to easily create animated nonlinear projections using traditional techniques of camera motion design. The set of example projection models developed here is exploratory and in no way exhaustive, and it remains as future work to extend these ideas and investigate their perceptual effects upon a person viewing the scene.

Chapter 6

Conclusions and Future Work

This thesis presents techniques that allow digital artists to interactively explore nonlinear projections of complex 3D scenes. These tools allow computer graphics artists and animators to work with a more general model of projection, one of the fundamental tools of depiction historically used by artists to manipulate a viewer's understanding of a scene. Building upon the idea of combining multiple linear perspective views, a comprehensive system for designing, animating, and rendering nonlinear projections has been developed. The use of multiple cameras to define these projections motivates the need for illumination models that account for the multiple vantage points. Constraints are introduced to allow for detailed control over scene composition and relative depth relationships among objects in a scene. The creative potential of this approach to developing nonlinear projections has been demonstrated by its extensive use in the production of the animated film *Ryan*, in which subtle manipulations of perspective are used to reveal the emotions of various characters.

As this approach is based on projection, it is inherently limited in that a given point in space can only project to one location in the image. If more general compositions are desired, multiple copies of the objects of interest could be created, each being projected as desired. Projections that fully unwrap a piece of 3D geometry can also be problematic;

a seam must be chosen for slicing the surface of the geometry. The nonlinearity of the projective mapping is also sensitive to resolution-dependent artifacts; automatic adaptive tessellation or splitting of geometry could be incorporated to reduce manual workflow requirements. Finally, while the system is very general and also very powerful, it has a large control space that can be complicated to work with, as users need to control not only the various cameras, but also each camera's associated weight functions and constraints.

To address the complexity of this system, a high-level approach to authoring nonlinear projections has been developed. Inspired by methods used by artists working with traditional 2D media, a direct specification interface allows users to edit 2D projections of 3D feature primitives on an image canvas to specify the overall desired projection. These desired projective mappings are cast as constraints in a nonlinear optimization problem, and acceptable solutions are used in conjunction with a heuristic algorithm for introducing local views to create the desired scene projection. Implicit weight functions associated with the feature primitives can then be edited to refine the scene projection beyond the constrained primitives.

However, the user must still manually edit viewport transformations when desired occlusion relationships do not result. Incorporating automatic preservation of existing occlusion relationships into the optimization problem could reduce the user's workflow. In addition, linear primitives are mapped to curved projections by partitioning the linear elements and blending weight functions at points of adjacency. Closer adherence to the desired projections could be achieved by using higher order interpolation of the weight functions of the partitioned linear elements. An alternative workflow worthy of investigation could incorporate the weight functions across all space into the optimization problem, using heuristics of low-distortion to derive weight functions throughout the 3D scene.

While the above approach allows digital artists to directly specify a nonlinear pro-

jection, it can be difficult to adapt to animated projections. Keyframed projections could be used, but the camera-feature relationship will generally be different for each projection, with even the number of cameras changing among the various projections. These complications preclude the automatic construction of intra-projection mappings using interpolated camera parameters, constraints and weight functions. One approach to solving this problem would involve defining interpolated projections as spatial interpolations of projected locations. However, it is not clear that such interpolations could be practically controlled. Beyond these technical complications, this approach is not easily amenable to the use of common cinematographic techniques, which are based upon the manipulation of a single, approximately linear perspective view.

To allow animators to use cinematographic principles, a set of procedural projection models has been developed that build upon a central animated camera. One example is a general-purpose hierarchical model, in which a single camera at the top of the hierarchy can be controlled by the user as is common in 3D computer animation. Additional cameras are added to the hierarchy as projective refinements over local regions of space, with each camera having its own associated constraints and weight functions. These lower level cameras, constraints, and weight functions are procedurally controlled within a given projection model, which can be controlled with intuitive parameters that are amenable to animation while producing a continuous projection over the parameter space. An example of a stylized projection model has been presented to illustrate the approach. In addition, two procedural projection models that adapt to camera motion and scene animation have been presented. These procedural approaches help alleviate the control problems of the direct manipulation technique, but requires the development of projection models to support a desired look. It remains as future work to further investigate procedural models of projection, including the possibility of developing a procedural model that could be keyframed with direct projection specification.

Bibliography

[1] *Time Slice Films Ltd. Chronology*. http://www.timeslicefilms.com/chronology_f.html, 2004.

[2] A. Adamson and V. Jensen. *Shrek*. Dreamworks Animation, 2001.

[3] Maneesh Agrawala, Denis Zorin, and Tamara Munzner. Artistic Multiprojection Rendering. In *Proceeding of the Eurographics Rendering Workshop 2000*, pages 125–136. ACM Press, 2000.

[4] Marc Alexa. Linear Combination of Transformations. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 380–387. ACM Press, 2002.

[5] S. Armstrong. The Nutcracket Suite. In *Fantasia*. Walt Disney Productions, 1940.

[6] Alan H. Barr. Global and Local Deformations of Solid Primitives. In *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques*, pages 21–30. ACM Press, 1984.

[7] Alan H. Barr. Ray Tracing Deformed Surfaces. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*, pages 287–296. ACM Press, 1986.

[8] Thaddeus Beier and Shawn Neely. Feature-Based Image Metamorphosis. In *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques*, pages 35–42. ACM Press, 1992.

[9] Jim Blinn. Where Am I? What Am I Looking At? *IEEE Computer Graphics and Applications*, 8(4):76–81, 1988.

[10] Veronique Bourgoin, Nathalie Farenc, and Marc Roelens. Creating Special Effects by Ray-Tracing with Non Classical Perspectives. Ecole Nationales des Mines de St. Etienne Technical Report 1995-13, 1995.

[11] M. S. T. Carpendale and Catherine Montagnese. A Framework for Unifying Presentation Space. In *Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology*, pages 61–70. ACM Press, 2001.

[12] S. Carpendale. *A Framework for Elastic Presentation Space*. PhD thesis, Simon Fraser University, 1999.

[13] David B. Christianson, Sean E. Anderson, Li wei He, David H. Salesin, Daniel S. Weld, and Michael F. Cohen. Declarative Camera Control for Automatic Cinematography. In *Proceedings of the AAAI '96*, 1996.

[14] Nelson Siu-Hang Chu and Chiew-Lan Tai. Animating Chinese Landscape Paintings and Panorama Using Multi-Perspective Modeling. In *Proceedings of CGI 2001*, 2001.

[15] Patrick Coleman and Karan Singh. RYAN: Rendering Your Animation Nonlinearly projected. In *Proceedings of the Third International Symposium on Non-Photorealistic Animation and Rendering*, 2004.

[16] J. P. Collomosse and P. M. Hall. Cubist Style Rendering from Photographs. *IEEE Transactions on Visualization and Computer Graphics*, 9(4):443–453, 2003.

[17] Julie O'B. Dorsey, François X. Sillion, and Donald P. Greenberg. Design and simulation of opera lighting and projection effects. In *Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques*, pages 41–50. ACM Press, 1991.

[18] Steven M. Drucker and David Zeltzer. CamDroid: A System for Implementing Intelligent Camera Control. In *Proceedings of the 1995 Symposium on Interactive 3D graphics*, pages 139–144. ACM Press, 1995.

[19] Bruno Ernst. *The Magic Mirror of M. C. Escher*. Evergreen, 2002.

[20] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hugues. *Computer Graphics: Principles and Practice*. Addison-Wesley, 1990.

[21] John Funge, Xiaoyuan Tu, and Demetri Terzopoulos. Cognitive Modeling: Knowledge, Reasoning and Planning for Intelligent Characters. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, pages 29–38. ACM Press/Addison-Wesley Publishing Co., 1999.

[22] Andrew Glassner. Cubism and Cameras: Free-form Optics for Computer Graphics. Microsoft Research Technical Report MSR-TR-2000-05, January 2000.

[23] Andrew Glassner. Digital Cubism. *IEEE Computer Graphics and Applications*, 24(3):82–90, 2004.

[24] Michael Gleicher and Andrew Witkin. Through-the-Lens Camera Control. In *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques*, pages 331–340. ACM Press, 1992.

[25] Michel Gondry. Like a rolling stone. 1995.

[26] Bruce Gooch, Erik Reinhard, Chris Moulding, and Peter Shirley. Artistic Composition for Image Creation. In *Proceedings of the Eurographics Workshop on Rendering*, 2001.

[27] Cindy Grimm. Post-Rendering Composition for 3D Scenes. In *Eurographics Short Papers 2001*, 2001.

[28] Eduard Gröller. Nonlinear Ray Tracing: Visualizing Strange Worlds. *The Visual Computer*, 11(5):263–276, 1995.

[29] L. Guterman. *Cats and Dogs*. Warner Bros. Studios, 2001.

[30] Richard Hartley and Rajiv Gupta. Linear Pushbroom Cameras. In *Proceedings of Third European Conference on Computer Vision*, pages 555–566, 1994.

[31] Li-wei He, Michael F. Cohen, and David H. Salesin. The Virtual Cinematographer: A Paradigm for Automatic Real-time Camera Control and Directing. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, pages 217–224. ACM Press, 1996.

[32] Alfred Hitchcock. *Vertigo*. Universal Studios, 1958.

[33] David Hockney. *Secret Knowledge*. Viking Press, 2001.

[34] Ron Howard. *Willow*. MGM Studios, 1988.

[35] Takeo Igarashi, Satoshi Matsuoka, and Hidehiko Tanaka. Teddy: A Sketching Interface for 3D Freeform Design. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, pages 409–416. ACM Press/Addison-Wesley Publishing Co., 1999.

[36] Scott Johnston. Non-photorealistic Rendering. In Anthony A. Apodaca and Larry Gritz, editors, *Advanced RenderMan: Creating CGI for Motion Pictures*. Morgan Kauffman, 1999.

[37] Steven D. Katz. *Film Directing Shot by Shot: Visualizing from Concept to Screen*. Michael Wiese Productions, 1991.

[38] Allison W. Klein, Tyler Grant, Adam Finkelstein, and Michael F. Cohen. Video Mosaics. In *Proceedings of the Second International Symposium on Non-Photorealistic Animation and Rendering*, pages 21–28. ACM Press, 2002.

[39] Allison W. Klein, Peter-Pike J. Sloan, Adam Finkelstein, and Michael F. Cohen. Stylized Video Cubes. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 15–22. ACM Press, 2002.

[40] Craig Kolb, Don Mitchell, and Pat Hanrahan. A Realistic Camera Model for Computer Graphics. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, pages 317–324. ACM Press, 1995.

[41] Chris Landreth. *Ryan*. 1380098 Ontario Inc. / the National Film Board of Canada, 2004.

[42] Seung-Yong Lee, Kyung-Yong Chwa, and Sung Yong Shin. Image Metamorphosis using Snakes and Free-form Deformations. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, pages 439–448. ACM Press, 1995.

[43] Jonathan Levene. A Framework for Non-Realistic Projections. Master's thesis, Massachusetts Institute of Technology, 1998.

[44] Helwig Löffelmann and Eduard Gröller. Ray Tracing with Extended Cameras. *The Journal of Visualization and Computer Animation*, 7(4):‘211–227, 1996.

[45] D. Martín, S. García, and J. C. Torres. Observer Dependent Deformations in Illustration. In *Proceedings of the First International Symposium on Non-Photorealistic Animation and Rendering*, pages 75–82. ACM Press, 2000.

[46] Nelson Max. Computer Graphics Distortion for IMAX and OMNIMAX Projection. In *Proceedings of Nicograph 1983*, pages 137–159, 1983.

[47] J. A. Nelder and R. Mead. A Simplex Method for Function Minimization. *Computer Journal*, 7:308–313, 1965.

[48] Leonard Nimoy. *Star Trek IV: The Voyage Home*. Paramount Productions, 1986.

[49] Shmuel Peleg, Benny Ruosso, Alex Rav-Acha, and Assaf Zomet. Mosaicing on Adaptive Manifolds. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(10):1144–1154, 2000.

[50] Paul Rademacher. View-Dependent Geometry. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, pages 439–446. ACM Press/Addison-Wesley Publishing Co., 1999.

[51] Paul Rademacher and Gary Bishop. Multiple-Center-of-Projection Images. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, pages 199–206. ACM Press, 1998.

[52] Thomas W. Sederberg and Scott R. Parry. Free-form Deformation of Solid Geometric Models. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*, pages 151–160. ACM Press, 1986.

[53] Steven Seitz and Jiwon Kim. The Space of All Stereo Images. *International Journal of Computer Vision*, 48(1):21–38, 2002.

[54] Steven Seitz and Jiwon Kim. Multiperspective Imaging. *IEEE Computer Graphics and Applications*, 23(6):16–19, 2003.

[55] Steven M. Seitz and Charles R. Dyer. View Morphing. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, pages 21–30. ACM Press, 1996.

[56] Heung-Yeung Shum and Li-Wei He. Rendering with Concentric Mosaics. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, pages 299–306. ACM Press/Addison-Wesley Publishing Co., 1999.

[57] Karan Singh. A Fresh Perspective. In *Proc. Graphics Interface 2002*, 2002.

[58] Dayton Taylor. Virtual Camera Movement: The Way of the Future? *American Cinematographer*, 17:93–100, 1996.

[59] Frank Thomas and Ollie Johnston. *The Illusion of Life: Disney Animation*. Disney Editions, 1981.

[60] Bill Tomlinson, Bruce Blumberg, and Delphine Nain. Expressive Autonomous Cinematography for Interactive Virtual Environments. In *Proceedings of the Fourth International Conference on Autonomous Agents*, pages 317–324. ACM Press, 2000.

[61] Scott Vallance and Paul Calder. Multi-Perspective Images for Visualization. In *Proceedings of the 2001 Pan-Sydney Area Workshop on Visual Information Processing*, 2001.

[62] Andy Wachowski and Larry Wachowski. *The Matrix*. Warner Bros. Studios, 1999.

[63] D. Weiskopf, T. Schafhitzel, and T. Ertl. GPU-Based Nonlinear Ray Tracing. In *Proceedings of Eurographics 2004*, 2004.

[64] Robert Wiene. *Das Kabinett des Doktor Caligari*. Image Entertainment, 1919.

[65] George Wolberg. *Digital Image Warping*. IEEE Computer Society Press, 1990.

[66] Daniel N. Wood, Adam Finkelstein, John F. Hughes, Craig E. Thayer, and David H. Salesin. Multiperspective Panoramas for Cel Animation. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, pages 243–250. ACM Press/Addison-Wesley Publishing Co., 1997.

[67] Geoff Wyvill and Craig McNaughton. Optical Models. In *Proceedings of CGI 1990*, 1990.

[68] Yonggao Yang, Jim X. Chen, Woosung Kim, and Changjin Kee. Nonlinear Projection: Using Deformations in 3D Viewing. *IEEE Computing in Science and Engineering*, 5(2):54–59, 2003.

[69] Jingyi Yu and Leonard McMillan. A Framework for Multiperspective Rendering. In *Proceedings of the 2004 Eurographics Symposium on Rendering*, 2004.

[70] Robert C. Zeleznik, Kenneth P. Herndon, and John F. Hughes. SKETCH: an Interface for Sketching 3D Scenes. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, pages 163–170. ACM Press, 1996.

[71] Denis Zorin and Alan H. Barr. Correction of Geometric Perceptual Distortions in Pictures. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, pages 257–264. ACM Press, 1995.