# IMAGE-BASED TRANSFORMATION OF VIEWPOINT AND SCENE APPEARANCE

By

**Steven Maxwell Seitz**

A dissertation submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
(Computer Sciences)

at the

**UNIVERSITY OF WISCONSIN – MADISON**

1997

# IMAGE-BASED TRANSFORMATION OF VIEWPOINT AND SCENE APPEARANCE

By

**Steven Maxwell Seitz**

A DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY
(COMPUTER SCIENCES)

at the

**UNIVERSITY OF WISCONSIN – MADISON**

1997

# Abstract

This thesis addresses the problem of synthesizing images of real scenes under three-dimensional transformations in viewpoint and appearance. Solving this problem enables interactive viewing of remote scenes on a computer, in which a user can move a virtual camera through the environment and virtually paint or sculpt objects in the scene. It is demonstrated that a variety of three-dimensional scene transformations can be rendered on a video display device by applying simple transformations to a set of basis images of the scene. The virtue of these transformations is that they operate directly on images and recover only the scene information that is required in order to accomplish the desired effect. Consequently, they are applicable in situations where accurate three-dimensional models are difficult or impossible to obtain.

A central topic is the problem of *view synthesis*, i.e., rendering images of a real scene from different camera viewpoints by processing a set of basis images. Towards this end, two algorithms are described that warp and resample pixels in a set of basis images to produce new images that are physically-valid, i.e., they correspond to what a real camera would see from the specified viewpoints. Techniques for synthesizing other types of transformations, e.g., non-rigid shape and color transformations, are also discussed. The techniques are found to perform well on a wide variety of real and synthetic images.

A basic question is uniqueness, i.e., for which views is the appearance of the scene uniquely determined from the information present in the basis views. An important contribution is a uniqueness result for the no-occlusion case, which proves that all views on the line segment between the two camera centers are uniquely determined from two uncalibrated views of a scene. Importantly, neither dense pixel correspondence nor camera information is needed. From this result, a *view morphing* algorithm is derived that produces high quality viewpoint and shape transformations from two uncalibrated images.

To treat the general case of many views, a novel *voxel coloring* framework is introduced that facilitates the analysis of ambiguities in correspondence and scene reconstruction. Using this framework, a new type of scene invariant, called *color invariant*, is derived, which provides intrinsic scene information useful for correspondence and view synthesis. Based on this result, an efficient voxel-based algorithm is introduced to compute reconstructions and dense correspondence from a set of basis views. This algorithm has several advantages, most notably its ability to easily handle occlusion and views that are arbitrarily far apart, and its usefulness for *panoramic* visualization of scenes. These factors also make the voxel coloring approach attractive as a means for obtaining high-quality three-dimensional reconstructions from photographs.

# Acknowledgements

This thesis is dedicated to my parents, Gary and Sheila Seitz, who taught me that education is a lifelong pursuit and inspired me to pursue a Ph.D.

I wish to thank Chuck Dyer for his guidance and support throughout my graduate studies. His advice and infallible judgement on all matters academic and otherwise have been a tremendous influence and an invaluable resource for me. I am very fortunate to have had the opportunity to work with him during my time at U.W. Madison. I would also like to thank the other members of my thesis committee, Nicola Ferrier, Vladimir Lumelsky, Vadim Shapiro, and Jude Shavlik for their comments and insights on this work.

I am especially grateful to Larry Rowe for introducing me to scientific research as an undergraduate at U.C. Berkeley and for encouraging me to pursue graduate studies in computer science. Kyros Kutulakos contributed key ideas to this work and provided a strong influence both as peer and collaborator.

In the course of graduate studies, I have benefited from interactions with many people. I would especially like to thank Brian Roddy, Aaron Seitz, Kathy Suchenski, Sanjay Tiwari, and John Watrous for many illuminating discussions. Advice and insights on this work were provided by several current and former members of the vision group at U.W. Madison, including Russel Manning, Brian Morgan, Andrew Prock, Dan Reznik, Brent Seales, and Steve Walden. I also thank the following people who graciously permitted their images to be used in this thesis, in order of appearance: John Watrous, Su-hui Chiang, Shubu Mukherjee, Dionisios Pnevmatikatos, Alain Kagi, Elton Glaser, Babak Falsafi, and Doug Burger.

Most of all, I am deeply grateful to Sandhya Subramanian for her endless encouragement and enthusiasm, and for making these last several years truly enjoyable.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Humans perceive the three-dimensional world by means of its projection onto our retinal fields. By mimicing this projection process, the eye can be fooled into perceiving depth and three-dimensionality in computer-generated imagery. Interactive computer graphics systems exploit this trick to enable a user to visualize and transform artificial objects and scenes in three dimensions via 2D display and input devices. In this thesis, we consider the problem of designing computer algorithms to visualize and manipulate the appearance of *real* objects and scenes that exist physically or have existed in the past. Unlike artificial scenes which must be painstakingly described in minute detail, real scenes have an intrinsic structure that can be visually captured in photographs and drawings.

At present, making use of this structure in computer graphics applications requires reconstructing the 3D structure of the scene from the images. This reconstruction problem, despite being well-studied in the computer vision literature, is not entirely solved and existing techniques are prone to a variety of errors and instabilities (cf. Chapter 6). Even relatively minor errors in shape can cause distracting artifacts when the model is texture-mapped and reprojected. Rather than attempt to extract a 3D scene description from the photographs, an alternative approach is to model the effects of scene transformations in the *image domain*, by determining how the image should change in response to a particular scene transformation. As shown in Fig. 1.1, applying a scene transformation and rendering the resulting scene yields a corresponding image transformation. Similarly, image transformations can *imply* scene transformations. By modifying the image, it should therefore be possible to mimic physical changes to the real 3D scene. We use the phrase *image-based scene transformation* to describe the synthesis of 3D scene, camera, or illumination transformations via modifications to one or more basis (input) images.

An advantage of this image-based approach is that precise modeling and reconstruction of shape, illumination, texture, and other scene quantities from images is avoided. Rather, we recover only the image information that is required to create the desired 3D effect. Consequently, image-based transformations can be simpler and more efficient than methods that require 3D reconstructions. More importantly, we will argue that image-based transformations are more *powerful* in that they enable visualization and manipulation of scenes that are difficult or impossible to reconstruct.

A transformation of particular interest is viewpoint (i.e., camera panning, zooming, and translation). Cast as an image transformation, a change in camera viewpoint can be modeled as a mapping, or *warp*, between pixels in one or more basis views and pixels in a new image, representing a synthetic view of the same scene. By categorizing the set of all such mappings, we can begin to devise

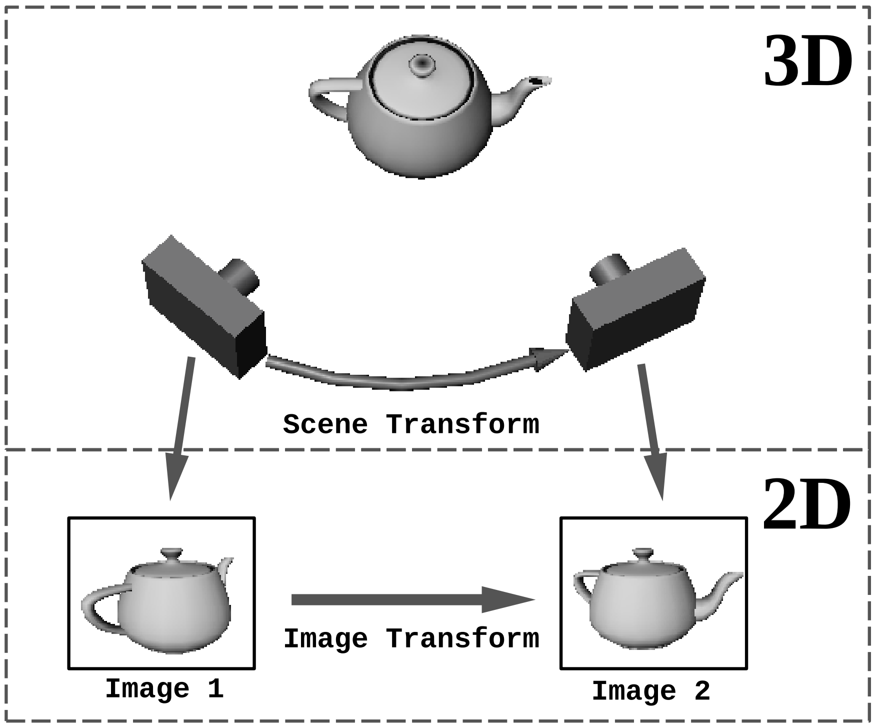


Figure 1.1: Image-based Scene Transformation. A scene transformation such as a change in camera viewpoint (top) causes the image to change (bottom). Similarly, a well-designed image transformation can *imply* a 3D transformation in the scene. We refer to this latter technique of synthesizing 3D effects via 2D image operations as *image-based scene transformation*.

mechanisms for *synthesizing* changes in viewpoint by warping images. This capability is critical for interactive graphics applications like visualization, virtual reality, flight simulation, and games, in which viewpoint changes continuously, subject to user-control. It also enables manipulation of scenes whose three-dimensional structure is unknown and unrecoverable. For instance, a present-day tourist could virtually visit a historic building, long ago destroyed, by warping a set of photographs from a time when the building was still standing. The user could view the building in 3D by moving a virtual camera through the environment. The illusion of a real three-dimensional space would be maintained by continually warping the images to correspond to the user's changing perspective. The same approach can be applied to drawings and paintings. For instance, Fig. 1.2 depicts a 3D rotation of Leonardo Da Vinci's *Mona Lisa*, generated using the image-based technique described in Chapter 3.

In this thesis we propose the use of image-based scene transformations as a general framework for visualizing and manipulating real 3D scenes on a computer display, and we describe algorithms to enable common transformations such as rigid and non-rigid 3D motions, viewpoint changes, and editing operations. These algorithms have the key advantage that they operate on a set of basis images

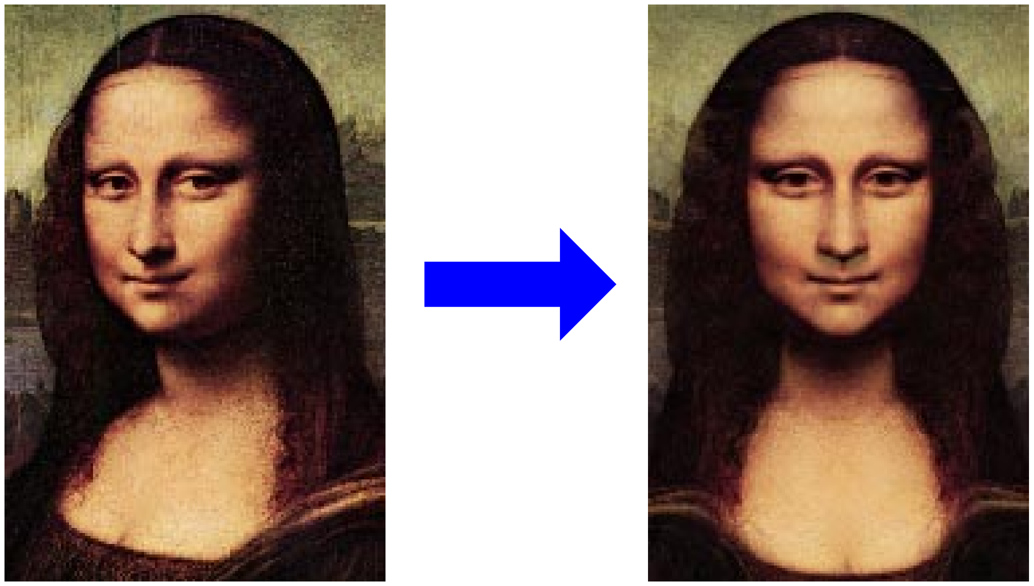


Figure 1.2: Image transformation of photographics, drawings, and paintings enables 3D visualization of historic objects and scenes. Here the Mona Lisa appears to rotate three-dimensionally by use of the view morphing technique described in Chapter 3.

and do not require 3D models of the scene. To further motivate the use of image transformations, we begin with examples of some simple image transformations that are capable of producing realistic 3D effects.

## 1.1 Examples of Image-Based Scene Transformations

In the early part of this century, Walt Disney and other pioneers developed the *cel animation* techniques that have been used since to generate cartoons and films like the *Three Little Pigs* [Wal33], *Fantasia* [Wal40a], and *Aladdin* [Wal92]. Cel animators employed a number of methods for conveying realistic motions, using paint and transparent sheets of celluloid as the basic medium. The success and continued popularity of their efforts attests to the efficacy of 2D techniques for producing realistic visual effects.

Whereas traditional animation focused primarily on 2D effects, many of which featured artistic distortions and exaggerations [Las87], certain 3D effects (i.e., *scene transformations*) were also employed. These included simulations of 3D camera translations by moving a small window over a background image. Translating the window conveys a frontoparallel motion of the camera, and scaling the window emulates forward or backward motion. While sufficient to give an impression of camera movement, this technique does not model depth effects like parallax and occlusion and therefore is

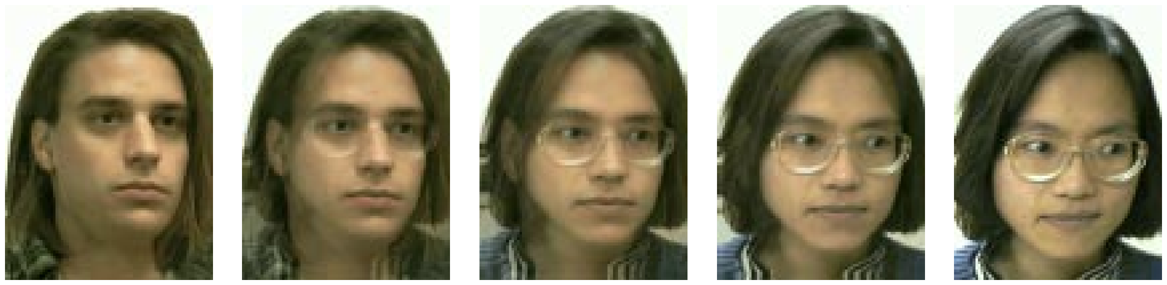


Figure 1.3: An image morph combines 2D interpolations of shape (via image warping) and color (via cross-dissolve) to create transitions that appear strikingly three-dimensional.

somewhat unrealistic. A more compelling effect can be achieved by designing the background image with a specific camera path in mind [WFH$^+$97]. This technique was used in Walt Disney's 1940 animated film, *Pinocchio* [Wal40b], in which a window was moved in a fixed path along a special background image. While the background appears warped when viewed in its entirety, it appears quite natural through the sliding window and the resulting animation gives a realistic 3D effect. Wood et al. [WFH$^+$97] have recently developed computer algorithms for partially automating the creation of these warped backgrounds, which they call *multiperspective panoramas*.

Another way of simulating depth effects in animation is through *layering*, or *trucking*, i.e., compositing a series of background images and windows that move at different speeds. This technique can model relative motion and occlusion between objects by representing the scene as a set of independently moving 2D layers rendered in back-to-front order. Recently, researchers in computer vision and computer graphics have advocated this layering paradigm as an effective way for representing and rendering more complicated 3D scenes by manually [TK96, LS97] or automatically [WA94, DP91, AS95] segmenting the scene into a series of layers.

In the last few years, computer-based image metamorphosis or *morphing* [Wol90, BN92] has emerged as a popular means for producing fascinating visual effects. These techniques combine 2D interpolations of shape and color to produce transitions between a pair of basis images. Generally, a morph is controlled by a set of correspondences and interpolation parameters that are user-provided and may be chosen to achieve effects such as the one shown in Fig. 1.3. Part of the appeal of morphing is that the images produced can appear strikingly lifelike and visually convincing. Despite being computed by 2D image transformations, effective morphs generate the illusion of a smooth 3D shape transformation between two objects. The fact that realistic 3D shape transformations can arise from 2D image morphs is rather surprising, but extremely useful, in that 3D shape modeling can be avoided.

While image morphs enable the synthesis of three-dimensional effects, they provide no direct control of the 3D transformation that is being synthesized. The morph in Fig. 1.3 clearly implies *some* three-dimensional shape transformation. But which one? Because the morph is user-specified in 2D, the corresponding 3D transformation is left implicit. As a result, simple 3D transformations such as viewpoint changes are often difficult to convey accurately with image morphing methods.

## 1.2 Transforming Real Scenes

The techniques in the previous section are examples where 2D representations and operations may be used to convey a strong impression of three-dimensionality. Because the primary goal of this thesis is transforming the appearance of *real* objects and scenes, we wish to develop similar image-based operations that can be applied to photographs and are sophisticated enough to precisely model viewpoint changes and other 3D operations.

Achieving these goals requires addressing a number of issues:

- **Measurability**: Sufficient information to compute the transformation must be automatically or semi-automatically extracted from the basis images.
- **Correctness**: Each synthesized image should be physically correct, i.e., it should correspond to what the real scene would look like as a result of the specified scene transformation.
- **Synthesis**: New algorithms must be developed for image-based scene transformations. The techniques should be robust, easy to use, and general enough to handle complex real-world objects and scenes.

Our focus will be synthesizing changes in camera position and orientation, i.e., *view synthesis*. We will also touch upon other types of transformations, specifically *editing* operations that modify surface shape and color, and describe how to perform these operations in the 2D domain.

## 1.3 Thesis Contributions

The main contribution of this thesis is a collection of results and algorithms that enable synthesizing 3D transformations of real scenes from basis images. In principle, this problem could be approached with a concatenation of steps that have been individually well-studied: (1) 3D reconstruction from images, (2) applying 3D scene, camera, and illumination transformations, and (3) rendering 3D scenes. However, closer examination of the problem reveals limitations of this approach, and that more powerful solutions are possible through the use of image transformations.

One important contribution is a uniqueness result that establishes, for the first time, that view synthesis is feasible without correspondence information or camera parameters. Specifically, given two views with no occlusions, it is possible to uniquely predict all in-between views along the line segment between the camera centers. This result is nontrivial because the *structure* of the scene is not uniquely recoverable given these conditions—shape reconstruction is ill-posed. The result is important because it paves the way for view synthesis algorithms that represent scene appearance with a collection of basis images. Based on this result, a second contribution is a novel technique called *view morphing* for synthesis of physically-correct, in-between views from a pair of uncalibrated images.

To generalize the two-view results, we also investigate the problem of view synthesis from numerous input images and relax the occlusion constraint. In this case the main challenge is computing pixel correspondences between images from cameras that may be far apart. We describe an algorithm that exploits a novel constraint on camera positions to provide an efficient solution to this correspondence problem. This algorithm has the unique feature of generating provably-consistent, dense correspondence maps from a set of input images, in the presence of occlusion. It is therefore useful not just for view synthesis but for other applications that require correspondence information, e.g., motion analysis and 3D scene reconstruction. This *voxel coloring* algorithm has several useful and novel features, including its ability to cope with widely spaced views and its capacity to provide *panoramic* scene visualizations and reconstructions.

An additional contribution is the *voxel coloring* framework for analyzing ambiguities and invariants in image correspondence and scene reconstruction. Using this framework, we identify certain points and correspondences with special invariant properties. Each such *color invariant* is a point having the same radiance distribution in every possible reconstruction of the scene from a given set of basis images, under fixed illumination. They therefore provide absolute radiance information that is constant across all consistent scenes. This is to be distinguished from methods that attempt to solve the *color constancy* problem [HSW92], by computing surface reflectance independent of scene illumination. In contrast, color invariants encode radiance directly, which is sufficient to synthesize new views of the scene with illumination held constant.

A common theme that underlies this work is the use of novel constraints that simplify problems that are otherwise intractable. The no-occlusion constraint, called *monotonicity*, is crucial for proving that view synthesis is tractable. Similarly, our solution for determining image correspondences from multiple views requires that the scene volume lies outside of the convex hull of all the camera centers. This constraint, which we call the *ordinal visibility constraint*, permits solving for correspondences in a generalized “depth-order.”

This thesis contains a combination of theoretical results and practical algorithms. These results require idealized assumptions (e.g., Lambertian surfaces) that are often violated in images of real scenes. However, the algorithms are much more general—they are robust to violations of the assumptions, and are shown to perform well on a wide range of real input sequences. When the assumptions are violated, the correctness proofs no longer apply and the results may not be entirely physically correct. However, for visualization applications, the primary goal is *perceptual realism*; physical correctness is only one means for achieving this goal and is not strictly necessary [Las87]. An important feature of these algorithms is that they produce realistic results even for significant violations of the assumptions. The view morphing algorithm in particular is shown to produce highly realistic viewpoint transitions in situations where the shading, visibility, and shape differ dramatically in the two input images. Moreover, the best results are obtained from images of faces and other natural objects that are extremely difficult to model using conventional methods, and for which the eye is highly sensitive to even minor errors in shape and color. These results suggest that the algorithms are in fact more widely applicable than the theory predicts.

## 1.4 Thesis Outline

The thesis begins by introducing terminology for describing color and spatial entities like scenes, images, and projections in Chapter 2.

Chapter 3 presents the problem of *view synthesis* and discusses the two-view case in detail. After an overview of related work, the problem is revisited from first principles in order to determine the conditions under which it is fully solvable. The first part of the chapter addresses these theoretical concerns and derives a uniqueness result for in-between views. The remainder of the chapter presents the *view morphing* algorithm for synthesizing in-between views from a pair of uncalibrated basis images.

The next two chapters investigate extensions of the view morphing algorithm for the cases of fewer than two and more than two basis views. Chapter 4 addresses the problem of view synthesis from a single view for objects with bilateral symmetry, and demonstrates its potential application for photo-correction and face recognition. Chapter 5 considers extensions to three or more views by cascading a sequence of pairwise view morphs.

Chapter 6 considers the problem of view synthesis from numerous input images that are widely distributed about a scene. The chapter begins with a discussion of the correspondence problem and reviews existing approaches to this basic problem. Motivated by limitations in the state of the art, a discrete scene-space framework is introduced, called *voxel coloring*. This framework is used to isolate and characterize fundamental ambiguities in the correspondence problem, and leads to the identification of points with special invariant properties. The collection of all such points, called *color invariants*, is shown to comprise a scene reconstruction that is fully consistent with the input images. An efficient algorithm is presented for computing these points from a set of basis images using a novel visibility constraint. The chapter concludes by evaluating the performance of the approach, applied to real and synthetic image sequences.

Chapter 7 discusses other 3D transformations and how they can be synthesized via image-based techniques. The focus is on editing operations, e.g., painting, scissoring, and morphing, that are applied in one image and propagate automatically to different views. After describing these *plenoptic image editing* operations, the chapter describes an implementation and presents results from a prototype system.

# Chapter 2

# Notation

## 2.1 Geometry

We write vectors and matrices in bold face and scalars in roman. Scene and image quantities are written in capitals and lowercase respectively. When possible, we also write corresponding image and scene quantities using the same letter. Images $\mathcal{I}$ and 3D shapes or scenes $\mathcal{S}$ are expressed as point sets. Following convention, we represent image and scene points using homogeneous coordinates. For example, an image point $[x\ y\ 1]^T = \mathbf{p} \in \mathcal{I}$ is the projection of a scene point $[X\ Y\ Z\ 1]^T = \mathbf{P} \in \mathcal{S}$. We reserve the notation $\mathbf{P}$ and $\mathbf{p}$ for points expressed in Euclidean coordinates whose last coordinate is 1. Scalar multiples of these points will be written with a tilde, as $\tilde{\mathbf{P}}$ and $\tilde{\mathbf{p}}$.

A camera is represented by a $3 \times 4$ homogeneous projection matrix of the form $\mathbf{\Pi} = [\mathbf{H} \mid -\mathbf{HC}]$. The vector $\mathbf{C}$ gives the position of the camera's optical center and the $3 \times 3$ matrix $\mathbf{H}$ specifies the transformation of its image plane with respect to the world coordinate system. The perspective projection equation is

$$\tilde{\mathbf{p}} = \mathbf{\Pi}\mathbf{P} \tag{2.1}$$

The term *view* will henceforth refer to the tuple $V = \langle \mathcal{I}, \mathbf{\Pi} \rangle$ comprised of an image and its associated projection matrix. We denote the image $\mathcal{I}$ of a scene $\mathcal{S}$ produced by a perspective camera at viewpoint $V$ by: $\mathcal{I} = P(\mathcal{S}, V)$.

## 2.2 Color

Image irradiance (pixel color) is given as a function $color(\mathbf{p}, \mathcal{I})$ mapping the position $\mathbf{p}$ of a pixel in image $\mathcal{I}$ to a point in color-space (e.g., monochrome intensity, RGB, HSV, etc.). The choice of color-space will be left unspecified unless needed for the purpose of discussion. For the case of Lambertian scenes, i.e., scenes containing only perfect matte surfaces, radiance is isotropic and can therefore be expressed using a scalar function. In this case, we define $color(\mathbf{P}, \mathcal{S})$ to be the radiance of a point $\mathbf{P}$ in scene $\mathcal{S}$.

# Chapter 3

# The Two-View Case: View Morphing

We now consider the problem of synthesizing images of a real scene from new camera viewpoints, by warping a pair of basis images. The problem is represented schematically in Fig. 3.1. This problem is interesting because (1) it has applications of practical importance, such as stereo viewing [MB95a] and teleconferencing [BP96], (2) it is amenable to a thorough bottom-up analysis, and (3) it provides a base case for the more general problem of view synthesis from arbitrary sets of views.

Towards this end, the first objective is to demonstrate that this view synthesis problem is indeed solvable, i.e., given two perspective views of a static scene, under what conditions may new views be unambiguously predicted? We point out that this question is nontrival, given that basic quantities like optical flow and shape are *not* uniquely computable due to inherent ambiguities (e.g., the aperture problem [Mar82]).

The second goal is to develop an algorithm to produce correct, high-quality, synthetic views of a scene from two basis images. The algorithm should produce correct views when the underlying assumptions are satisfied, yet be sufficiently robust to cope with large deviations, e.g., non-static scenes or varying illumination.

## 3.1 2D vs. 3D

Can the appearance from new viewpoints of a static three-dimensional scene be predicted from a set of basis views of the same scene? One way of addressing this question is to consider view synthesis as a two-step procedure—reconstruct the scene from the basis views using stereo or structure from motion methods and then reproject to form new views. Indeed, a great deal of recent work [TK92, BTZ96, Sze96, MKKJ96, KRN97] has been devoted to the problem of obtaining high-quality texture-mapped 3D models from two or more basis images, and significant progress has been made in this area.

The problem with this two-step paradigm is that view synthesis becomes at least as difficult as 3D scene reconstruction. This conclusion is especially unfortunate in light of the fact that 3D reconstruction from photographs is generally ambiguous—a number of different scenes may be consistent with a given set of images; it is an ill-posed problem [PTK85]. This suggests that view synthesis is also ill-posed. However, it can be shown that this ambiguity is an artifact of 3D reconstruction and is avoidable for the task of synthesizing new 2D views. By extracting only the information needed for view

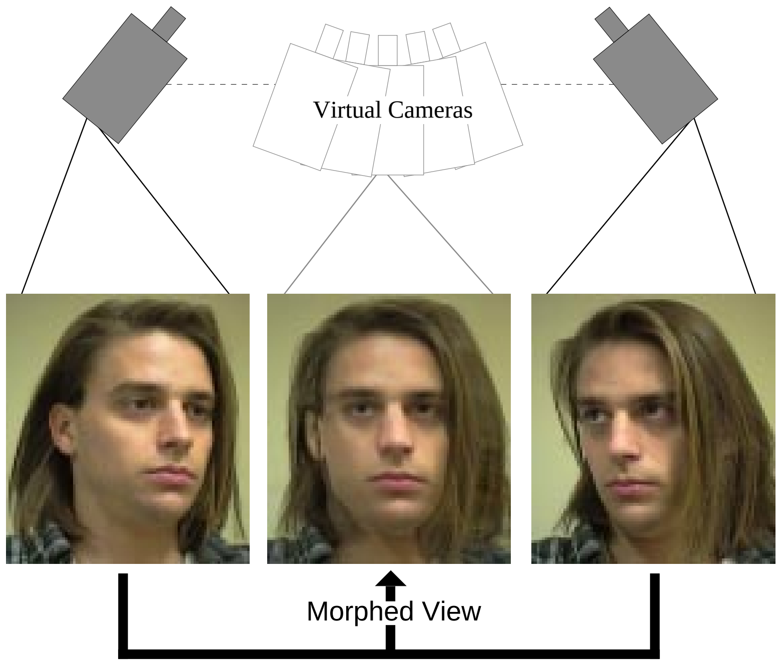


Figure 3.1: View morphing between two images of an object taken from two different viewpoints produces the illusion of physically moving a virtual camera.

synthesis, well-known limitations like the *aperture problem* [Mar82] are eliminated. In additional to these fundamental limitations, there are practical problems with the reconstruction approach— existing reconstruction methods are prone to errors due to occlusion and surface discontinuity that lead to distracting visible artifacts in synthesized views.

Even in cases where good reconstructions are computable, there are strong reasons for favoring a 2D, *image-based* approach to view synthesis. Note that the view synthesis problem is entirely two-dimensional—the input and output are comprised of 2D images. Rather than try to build a 3D model, a more direct approach is to compute a 2D to 2D mapping that rearranges pixels in the input images to synthesize an output image. By keeping the problem in the image domain, the complexity and error accumulation incurred by multiple 2D to 3D and 3D to 2D transformations are avoided. These arguments motivate using 2D image-based scene representations for view synthesis applications.

In spite of these reasons for favoring image-based representations, for the case of many input images, 3D models can offer a more compact scene representation. Observe that for $N$ input images, a naive image-based approach would store all $N$ images and $N^2$ correspondence maps, resulting in an expensive and highly redundant representation. As the number of images becomes very large, 3D or even 4D [LH96, GGSC96] representations can offer distinct advantages. A new solution for large numbers of images is presented in Chapter 6.

## 3.2 Related Work on View Synthesis

Lippman [Lip80] was one of the first to propose the use of an image-based representation for computer-aided 3D scene visualization. His *Movie-Map* approach enabled interactive virtual exploration of an environment by selectively accessing views that were previously captured and stored on laser disk. An additional contribution was the use of panoramic imaging to capture a wide field of view. These images were optically corrected at playback time using a viewing apparatus employing a conical mirror. By moving one's head relative to this device, the user could effect a rotation of the virtual camera about its optical center.

In their seminal paper on view synthesis, Chen and Williams [CW93], described how new views of a scene could be quickly computed from a set of basis images and range-maps via image warping in software on desktop computers. A primary motivation was speed; interpolation could be performed more quickly than rendering directly from a 3D model, especially when high quality rendering methods like ray tracing or radiosity were used. This observation led to the development of novel hardware systems that use view synthesis techniques to achieve real-time rendering rates for synthetic 3D scenes [RP94, TK96, LS97].

Laveau and Faugeras [LF94] were among the first to develop view synthesis techniques that operate on photographs and therefore apply to real scenes. While others [TK92, OLC93] had devised methods for synthesizing views from known (a priori or derived) depth maps and camera positions, they were the first to demonstrate the feasibility of view synthesis directly from uncalibrated images and correspondence maps. To solve the problem of mapping points between uncalibrated views, they developed techniques based on the *fundamental matrix* [LH81, DZLF94]. A second contribution was a ray-tracing method for resolving occlusion relationships in synthesized views.

Poggio and Brunelli [PB92] proposed a computational learning framework for synthesizing 2D and 3D transformations from a set of basis images. Their technique, which uses hyper basis functions to learn parameterized models from images, is very general and can model a variety of interesting transformations. Later work [BP96, VP96] demonstrated the utility of this framework for the difficult case of synthesizing new views of human faces. An important contribution of this work was its exceptionally broad scope—while a primary focus is view synthesis, their learning framework is equally useful for modeling changes in facial expression and other non-rigid transformations, and is effective both for synthesis *and* recognition applications.

The last three years has seen an explosion in interest in view synthesis techniques and applications. Concurrent with the work in this thesis [SD95b, SD96c, SD96b, Sei97, SD97d, SD97a, SD97b, SD97c], numerous other researchers [MP95, MP97, WHH95, IAH95, KNR95, SK95, KAI$^+$95, MB95a, MB95b, Che95, KTOT95, Sze96, MKKJ96, BP96, Sch96, VP96, DTM96, GGSC96, LH96, AS97, Rob97, HiAA97, SS97, KRN97] have developed other view synthesis techniques. For brevity, we discuss only a few here that are representative of the dominant approaches that currently exist. Comparisons with approaches in this thesis will be discussed in later sections and chapters.

Chen [Che95], and McMillan and Bishop [MB95b] developed systems similar to Lippman's Movie-Maps but using computer-based rather than optical methods. Both of these systems provided panoramic visualization of scenes using one or more cylindrical image mosaics, each of which stored scene appearance from a single camera position. McMillan and Bishop's *Plenoptic Modeling* approach also supported interpolation of mosaics to allow camera translation using an elegant image-space algorithm to resolve visibility order. Apple Computer's commercialization of Chen's system, *QuickTime VR* [App95], has brought image-based scene visualization to the mainstream, enabling interactive viewing on conventional PC's of real scenes like the Great Wall of China or the interior of a new car. The success of QuickTime VR and newer systems like *Surround Video* [Bla97], *IPIX* [Int97], *Smooth-Move* [Inf97], and *RealVR* [Liv97] has helped spawn an emerging subfield of computer graphics, often called *image-based rendering*, and has attracted growing interest in image-based representations.

Debevec et al. [DTM96] exploited a mixture of manual and automatic methods to reconstruct high-quality 3D models of architectural scenes. A view-dependent texture mapping approach was used to model varying illumination and reflectance. A major contribution of this work was to show the importance of user-interaction to aid view synthesis algorithms and facilitate 3D model construction.

Levoy and Hanrahan [LH96] and Gortler et al. [GGSC96] developed novel ray-based approaches for view synthesis. These methods, termed *light field* and *Lumigraph* respectively, represent the visible scene using a four-dimensional ray space in which any image is a two-dimensional sample. The problems of acquiring the representation and synthesizing views are both achieved via simple sampling operations. While elegant and relatively simple to construct, these approaches require many more input views compared to correspondence-based techniques.

Most closely-related to the approach in this chapter are the so called *view interpolation* methods [CW93, LF94, MB95b, WHH95, BP96, Sch96, AS97], which use image warping to produce new views from a small number of basis views. Most of these methods require advance knowledge of camera parameters to produce correct perspective views, with the exceptions of [LF94, AS97]. Furthermore, all of these methods require dense pixel correspondence maps as input. This latter requirement is a serious limitation, given that image-based computation of correct correspondence maps is known to be an ill-posed problem [PTK85, VP89]. In contrast, this chapter presents a view interpolation technique that operates directly on a pair of images, without knowledge of camera parameters or dense correspondence. It has the unique property of requiring only *measurable* image information, thus establishing that view interpolation is a well-posed problem.

While the method developed in this chapter can operate fully automatically, a key feature is the use of user-interaction to help specify new views and control the interpolation. In this respect, our approach has strong similarities to Debevec et al.'s [DTM96] *Facade* system which included an important manual component. However, the approach taken in this chapter is very different from *Facade*; the former produces new views via 2D image warping operations, whereas the latter does so by constructing texture-mapped 3D models.

## 3.3 Issues

We now consider the view synthesis problem from first principles, so as to determine the theoretical feasibility of synthesizing new views of a scene from a pair of basis images. Specifically, the following issues must be resolved:

- **Uniqueness**: for the problem to be well-posed, scene appearance from the new viewpoint must be *uniquely determined* from the basis images. The new view must be constructed entirely from *measurable* image information.
- **Correctness**: the synthesized image should be physically correct, i.e., it should correspond to what a real camera would see from the chosen viewpoint.
- **Registration**: the input images must be registered to determine pixel correspondence and quantities relating to the underlying camera geometry (e.g., epipolar lines). The problem must be formulated in a way that avoids known limitations on what can be measured from two images. In particular, complete pixel correspondence and Euclidean camera positions should not be required.
- **Uncalibrated Cameras**: the method should be applicable in cases where the images are *uncalibrated*, i.e., when camera positions and parameters are unknown.
- **Synthesis**: new algorithms must be developed that synthesize correct, high quality views from a pair of basis images. The techniques should be robust, easy to use, and general enough for application to faces and other objects and scenes for which construction of realistic 3D models is difficult.

These issues are addressed in the remainder of this chapter. We begin by analyzing the uniqueness issue and considering the conditions under which the view synthesis problem is theoretically solvable. This leads to the question of *measurability*, i.e., what information must be extracted from the basis images in order to solve the problem. Section 3.6 reviews the technique of image morphing and discusses its potential applicability for view synthesis. After analyzing some of its shortcomings, we introduce a modification, called *view morphing*, that is better suited for view synthesis. The technique is first described for the case where the camera positions are known and is then extended in Section 3.10 to work for images from unknown, uncalibrated cameras. The chapter concludes with several experiments, demonstrating the effectiveness of the approach for a variety of different image pairs.

## 3.4 Uniqueness and Measurability

Before developing view synthesis algorithms, it must first be demonstrated that the problem is solvable. Namely, is there sufficient information in two basis views of a scene to uniquely predict how

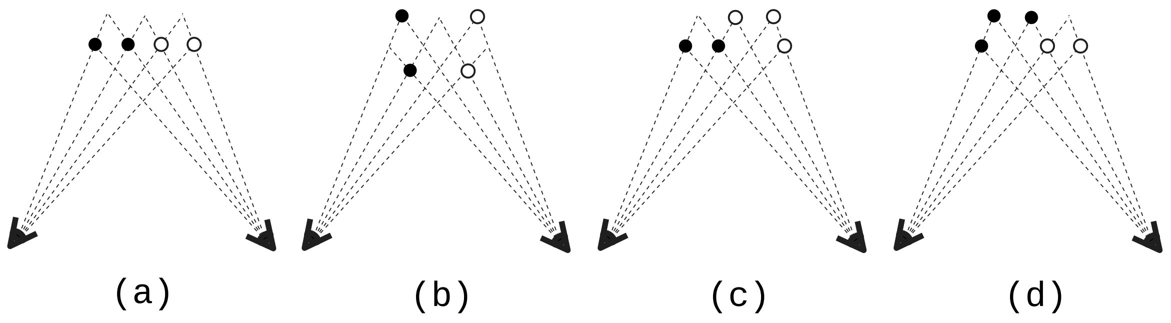


Figure 3.2: Visual Ambiguity. All four scenes are indistinguishable from these two viewpoints, but appear differently from other camera positions. Therefore, scene appearance is not uniquely determined from these two basis images.

that scene appears from new viewpoints? Fig. 3.2 illustrates the difficulty: a pair of images could be consistent with several different scenes, all of which appear indistinguishable from these two viewpoints, but distinct from other viewpoints. Given a pair of images of a scene, it is therefore, in general, impossible to predict how that particular scene would appear from new viewpoints. Despite the lack of a completely general solution, the following fundamental questions can be posed:

- **Uniqueness**: Under what conditions are new views uniquely determined from two basis images, and which views are determined?
- **Measurability**: What information must be extracted from the basis images in order to correctly synthesize these views?

Answering these questions is crucial to understanding the applicability and limitations of view synthesis algorithms. In this chapter we take significant steps toward this goal by demonstrating that a specific range of perspective views is uniquely determined from two or more basis views under a generic visibility assumption called *monotonicity*. Importantly, it is shown that generating these views relies entirely on *measurable* image information, avoiding ill-posed correspondence problems entirely. Chapter 5 extends these results for the case of more than two views. To our knowledge, these results are the first to prove the feasibility of view synthesis without correspondence or 3D shape information.

## 3.5 Monotonicity

In this section we consider the conditions under which new views of a scene may be uniquely predicted from a pair of basis images. In order to proceed, we must first identify what information must be extracted from the basis images to predict scene appearance from new viewpoints. It is useful to consider this problem within the framework of optical flow. For any pair of images $\mathcal{I}_0$ and $\mathcal{I}_1$ of the same scene, there exist *optical flow* fields $C_{01} : \mathcal{I}_0 \Rightarrow \mathcal{I}_1$ and $C_{10} : \mathcal{I}_1 \Rightarrow \mathcal{I}_0$, specifying the correspondence of projected scene points in the two images. These maps are defined by the relation:

$$C_{ij}(\mathbf{p}_i) = \mathbf{p}_j \quad \text{if } \mathbf{p}_i \in \mathcal{I}_i \text{ and } \mathbf{p}_j \in \mathcal{I}_j \text{ are projections of the same scene point}$$

The optical flow is undefined at points which are occluded (not visible) in one of the two images. We say a flow field is complete if it is bijective and defined at every image pixel.

When a flow field is complete, it can be used for view synthesis. Given an image $\mathcal{I}_0$ of a Lambertian scene and a complete flow field $C_{s0}$, we can synthesize a new image $\mathcal{I}_s$ by the relation:

$$color(\mathbf{p}, \mathcal{I}_s) = color(C_{s0}(\mathbf{p}), \mathcal{I}_0)$$

The view synthesis problem can therefore be reduced to the problem of determining flow fields. This approach is in essence the reduction used in all previous approaches to view synthesis, for instance [CW93, LF94, MB95b, BP96]. While complete optical flow is sufficient to synthesize new views, its automatic recovery from basis images is problematic and error-prone. In particular, the flow within low-contrast regions of nearly uniform color is locally ambiguous, a condition known as the *aperture problem* [Mar82]. As a result, complete optical flow is generally not measurable from two images [PTK85, VP89]. In response to the aperture problem many researchers instead use *normal flow* [Hil83], which provides only the motion component in the gradient direction but has the advantage of being measurable. In this section we introduce another measurable variety of flow called *boundary flow* that is particularly well-suited to synthesizing new views of a scene. Its derivation relies upon a visibility constraint called *monotonicity*.

The motivation behind boundary flow is that view synthesis is possible with only a subset of the optical flow field. Optical flow estimates become ill-conditioned within homogeneous regions of uniform color, but can be reliably computed at the boundaries of these regions, i.e., at image discontinuities. Note, however, that predicting the boundary is sufficient to predict the appearance of such a region in a new view, since its interior is known to be uniform. Intuitively, a red “blob” in one image will appear as a red blob in a second image; one need only determine the shape of the blob. This argument hinges on the notion that uniform regions are “preserved” in different views, a constraint formalized by the condition of *monotonicity*, which is introduced next.

Consider two views, $V_0$ and $V_1$, with respective optical centers $\mathbf{C}_0$ and $\mathbf{C}_1$, and images $\mathcal{I}_0$ and $\mathcal{I}_1$. Denote $\overline{\mathbf{C}_0\mathbf{C}_1}$ as the line segment connecting the two optical centers. Any point $\mathbf{P}$ in the scene determines an epipolar plane containing $\mathbf{P}$, $\mathbf{C}_0$, and $\mathbf{C}_1$ that intersects the two images in conjugate epipolar lines. The monotonicity constraint dictates that all visible scene points appear in the same order along conjugate epipolar lines of $\mathcal{I}_0$ and $\mathcal{I}_1$. This constraint is used commonly in stereo matching [OK85a] because the fixed relative ordering of points along epipolar lines simplifies the correspondence problem. Despite its usual definition with respect to epipolar lines and images, monotonicity constrains only the location of the optical centers with respect to points in the scene—the image planes may be chosen arbitrarily. An alternate definition that isolates this dependence more clearly is shown

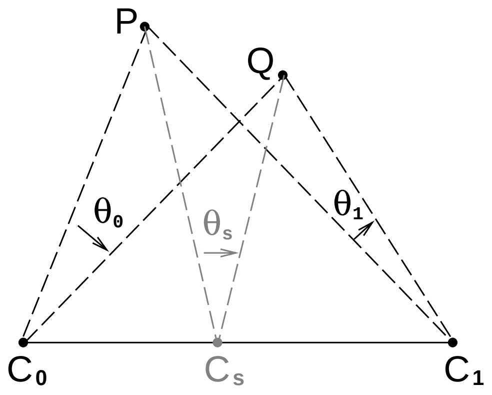


Figure 3.3: The Monotonicity Constraint. Any two points $\mathbf{P}$ and $\mathbf{Q}$ in the same epipolar plane determine angles $\theta_0$ and $\theta_1$ with the respective camera optical centers, $\mathbf{C}_0$ and $\mathbf{C}_1$. For monotonicity to apply, these angles must satisfy $\theta_0\theta_1 > 0$. If satisfied for $\mathbf{C}_0$ and $\mathbf{C}_1$, monotonicity applies as well for any other view with optical center along $\overline{\mathbf{C}_0\mathbf{C}_1}$.

in Fig. 3.3. Any two scene points $\mathbf{P}$ and $\mathbf{Q}$ in the same epipolar plane determine angles $\theta_0$ and $\theta_1$ with the optical centers $\mathbf{C}_0$ and $\mathbf{C}_1$. The monotonicity constraint dictates that for all such points $\theta_0$ and $\theta_1$ must be nonzero and of equal sign. The fact that no constraint is made on the image planes is of primary importance for view synthesis because it means that *monotonicity is preserved under homographies*, i.e., under image reprojection. This fact will be essential in the next section for developing an algorithm for view synthesis.

### 3.5.1 Impact for View Synthesis

A useful consequence of monotonicity is that it extends to cover a continuous range of views in-between $V_0$ and $V_1$. We say that a third view $V_s$ is *in-between* $V_0$ and $V_1$ if its optical center $\mathbf{C}_s$ is on $\overline{\mathbf{C}_0\mathbf{C}_1}$. Observe that monotonicity is violated only when there exist two scene points $\mathbf{P}$ and $\mathbf{Q}$ in the same epipolar plane such that the infinite line $\mathbf{PQ}$ through $\mathbf{P}$ and $\mathbf{Q}$ intersects $\overline{\mathbf{C}_0\mathbf{C}_1}$. But $\mathbf{PQ}$ intersects $\overline{\mathbf{C}_0\mathbf{C}_1}$ if and only if it intersects either $\overline{\mathbf{C}_0\mathbf{C}_s}$ or $\overline{\mathbf{C}_s\mathbf{C}_1}$. Therefore monotonicity applies to in-between views as well, i.e., signs of angles are preserved and visible scene points appear in the same order along conjugate epipolar lines of all views along $\overline{\mathbf{C}_0\mathbf{C}_1}$. We therefore refer to the range of views with centers on $\overline{\mathbf{C}_0\mathbf{C}_1}$ as a *monotonic range* of viewspace. Notice that this range gives a lower bound on the range of views for which monotonicity is satisfied in the sense that the latter set contains the former. For instance, in Fig. 3.3 monotonicity is satisfied for all views on the open ray from the point $\mathbf{C}_0\mathbf{C}_1 \bigcap \mathbf{PQ}$ through both camera centers. However, without *a priori* knowledge of the geometry of the scene, we may infer only that monotonicity is satisfied for the range $\overline{\mathbf{C}_0\mathbf{C}_1}$.

The property that monotonicity applies to in-between views is quite powerful and is sufficient to completely predict the appearance of the visible scene from all viewpoints along $\overline{\mathbf{C}_0\mathbf{C}_1}$. Consider the projections of a set of uniform Lambertian surfaces (each surface has uniform radiance, but any two surfaces may have different radiances) into views $V_0$ and $V_1$. Fig. 3.4 shows cross sections $S_1$, $S_2$,

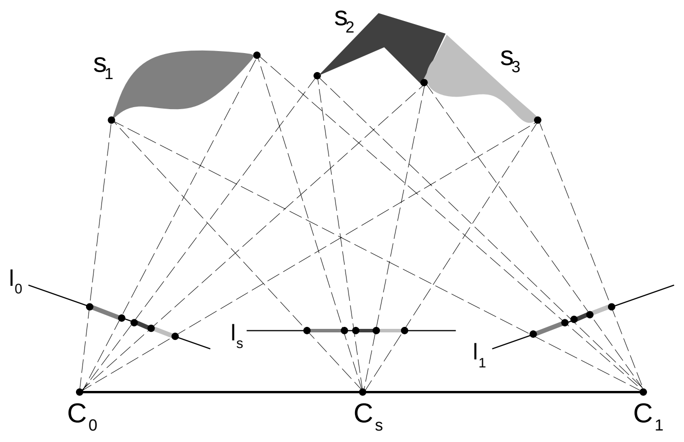


Figure 3.4: Correspondence Under Monotonicity. Cross-section view of three surfaces projecting into conjugate epipolar lines of three images. Although the projected intervals in $l_0$ and $l_1$ do not provide enough information to reconstruct $S_1$, $S_2$, and $S_3$, they are sufficient to predict the appearance of $l_s$.

and $S_3$ of three such surfaces projecting into conjugate epipolar lines $l_0$ and $l_1$. Each connected cross section projects to a uniform interval (i.e., an interval of uniform color) of $l_0$ and $l_1$. The monotonicity constraint induces a correspondence between the endpoints of the intervals in $l_0$ and $l_1$, determined by their relative ordering. The points on $S_1$, $S_2$, and $S_3$ projecting to the interval endpoints are determined from this correspondence by triangulation. We will refer to these scene points as *visible endpoints* of $S_1$, $S_2$, and $S_3$. We use the term *boundary flow* in reference to the mapping between endpoints of uniform intervals in $l_0$ and $l_1$. This is the subset of the optical flow field defined at interval endpoints, and is measurable by the above argument.

Now consider an in-between view, $V_s$, with image $\mathcal{I}_s$ and corresponding epipolar line $l_s$. As a consequence of monotonicity, $S_1$, $S_2$, and $S_3$ project to three uniform intervals along $l_s$, delimited by the projections of their visible endpoints. Notice that the intermediate image does not depend on the specific shapes of surfaces in the scene, only on the positions of their visible endpoints. Any number of distinct scenes could have produced $\mathcal{I}_0$ and $\mathcal{I}_1$, but each one would also produce the same set of intermediate images. More formally,

**Uniqueness of In-Between Views**: Let $\mathcal{I}_0$ and $\mathcal{I}_1$ satisfy monotonicity and let $\aleph$ be the set of all scenes $\mathcal{S}$ such that $P(\mathcal{S}, V_0) = \mathcal{I}_0$ and $P(\mathcal{S}, V_1) = \mathcal{I}_1$. For any two scenes $\mathcal{S}, \mathcal{S}' \in \aleph$ and in-between view $V_s$, $P(\mathcal{S}, V_s) = P(\mathcal{S}', V_s)$.

Hence, all views along $\overline{\mathbf{C}_0\mathbf{C}_1}$ are determined from $\mathcal{I}_0$ and $\mathcal{I}_1$. This result demonstrates that view synthesis under monotonicity is an intrinsically well-posed problem—and is therefore much easier than 3D reconstruction and related motion analysis tasks requiring smoothness conditions and regularization techniques [PTK85].

To see why the monotonicity constraint is so crucial to view synthesis, observe that it is required not only to make the correspondence problem solvable, but also to predict the appearance of uniform surfaces whose shapes are unknown. Furthermore, the in-between views are the *only* views that can be predicted with certainty due to the requirement that the visible endpoints of each surface remain fixed.

The monotonicity constraint is closely related to the theory of aspect graphs and visual events [KvD76, KvD79]. The constraint dictates that no changes in visibility, i.e., no occlusions, may occur within a monotonic range of viewspace. In other words, all views within a monotonic range are topologically equivalent; the same scene points are visible in every view. This condition of photometric topological equivalence is somewhat stronger than the notion of topological equivalence of image contour structure used to define an aspect. Consequently, a monotonic range of viewspace always occurs within an aspect.

### 3.5.2 Identifying Monotonic Scenes

A final question concerns the *measurability of monotonicity*. Under the monotonicity assumption we have established that view synthesis is feasible and relies only on measurable image correspondence information. However we have not yet considered whether or not monotonicity itself is measurable—can we determine if two images of a scene satisfy monotonicity by inspecting the images themselves or must we know the answer a priori? Strictly speaking, monotonicity is not measurable, in the sense that two images may be consistent with multiple scenes, some of which satisfy monotonicity and others that do not. However, we may determine whether or not two images are *consistent* with a scene for which monotonicity applies, by checking that each epipolar line in the first image is a monotonic warp of its conjugate in the second image. That is, if $l_0$ and $l_1$ are conjugate epipolar lines, expressed as functions mapping position to color, there exists a monotonic function $\sigma$ such that $l_0 = l_1 \circ \sigma$. If we denote by $\mathcal{M}$ the class of all monotonic scenes consistent with two basis images, this consistency property says that we may determine from the basis images whether or not $\mathcal{M}$ is empty. If $\mathcal{M}$ is nonempty, the result of view synthesis is a set of images that are consistent with every scene in $\mathcal{M}$.

### 3.5.3 Orthographic Views

For pairs of orthographic views, monotonicity cannot be defined with respect to camera centers, as above. However, the same results apply if we instead define monotonicity in terms of the ordering of points along conjugate epipolar lines. In this section we summarize the results for the orthographic case. Details can be found in Appendix B.

In the orthographic case, monotonicity states that the projections of any two points $\mathbf{P}$ and $\mathbf{Q}$ in the same epipolar plane appear in the same order along conjugate epipolar lines $l_0$ and $l_1$. If this property holds for all corresponding epipolar lines in the two views then we say that monotonicity holds for $V_0$ and $V_1$. Let $\mathbf{N}_i$ denote the image plane unit normal of $V_i$, also known as the *optical axis* or *direction of gaze*. Geometrically, the monotonicity constraint dictates that the line through $\mathbf{P}$ and $\mathbf{Q}$ may not intersect the line segment $\overline{\mathbf{N}_0\mathbf{N}_1}$ joining the tips of the two image plane normals (see Fig. B.1).

As in the perspective case, monotonicity extends to cover a range of views in-between $V_0$ and $V_1$. We say that a third view $V_s$ is *in-between* $V_0$ and $V_1$ if its normal $\mathbf{N}_s$ intersects $\overline{\mathbf{N}_0\mathbf{N}_1}$. The uniqueness result holds for orthography as well: $\mathcal{I}_s$ is uniquely determined from $\mathcal{I}_0$ and $\mathcal{I}_1$.

## 3.6 Image Morphing for View Synthesis

In attempting to develop view synthesis algorithms, we were inspired by the realistic effects obtained by *image morphing* techniques. These techniques combine 2D interpolations of shape and color to create seamless and often dramatic transitions between a pair of input images. Fig. 1.3 provides an example of an image morph between two faces. Part of the appeal of morphing is that the images produced can appear strikingly lifelike and visually convincing, as seen in the figure. Despite being computed by 2D image transformations, effective morphs can suggest a natural transformation between objects in the 3D world.

Given that 2D image morphs can effectively convey 3D transformations, it is natural to consider if image morphing can be used to synthesize new views of a scene. Specifically, suppose we morphed images of an object from two different camera positions using an existing morphing program. Would the resulting sequence of in-between images correspond to correct perspective views of the scene? In order to answer this question, we must first consider some basic principles governing how current image morphing techniques work.

A morph is determined from two images $\mathcal{I}_0$ and $\mathcal{I}_1$ and maps $C_{01} : \mathcal{I}_0 \Rightarrow \mathcal{I}_1$ and $C_{10} : \mathcal{I}_1 \Rightarrow \mathcal{I}_0$ specifying a complete correspondence between points in the two images. Two maps are required because the correspondence may not be one-to-one. In practice, $C_{01}$ and $C_{10}$ are partially specified by having the user provide a sparse set of matching features or regions in the two images. The remaining correspondences are determined automatically by interpolation [Wol90, BN92, LCSW92]. A warp function for each image is computed from the correspondence maps, usually based on linear interpolation:

$$C_{0s}(\mathbf{p}_0) = (1-s)\mathbf{p}_0 + sC_{01}(\mathbf{p}_0) \tag{3.2}$$

$$C_{1s}(\mathbf{p}_1) = (1-s)C_{10}(\mathbf{p}_1) + s\mathbf{p}_1 \tag{3.3}$$

$C_{0s}$ and $C_{1s}$ give the displacement of each point $\mathbf{p}_0 \in \mathcal{I}_0$ and $\mathbf{p}_1 \in \mathcal{I}_1$ as a function of $s \in [0,1]$. The in-between images $\mathcal{I}_s$ are computed by warping the two original images and averaging the pixel colors of the warped images. Existing morphing methods vary principally in how the correspondence maps are computed. In addition, some techniques allow finer control over interpolation rates and methods. For instance, Beier and Neely [BN92] suggested two different methods of interpolating line features, using linear interpolation of endpoints, per Eqs. (3.2) and (3.3), or of position and angle. In this thesis, the term *image morphing* refers specifically to methods that use pointwise linear interpolation to compute feature positions in in-between images, including [Wol90, BN92, LCSW92].

To illustrate the potentially severe 3D distortions incurred by image morphing, it is useful to consider interpolating between two different views of a planar shape. Any two such images are related by

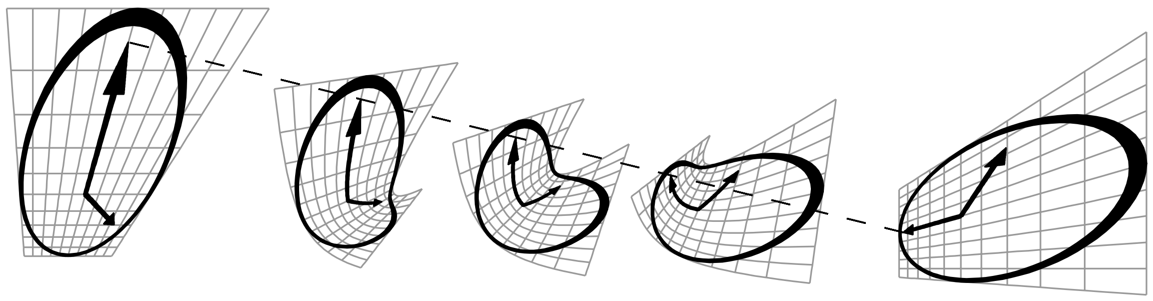


Figure 3.5: A Shape-Distorting Morph. Linearly interpolating two perspective views of a clock (far left and far right) causes a geometric bending effect in the in-between images. The dashed line shows the linear path of one feature during the course of the transformation. This example is indicative of the types of distortions that can arise with image morphing techniques.

a 2D projective mapping of the form:

$$H(x,y) = (\frac{ax+by+c}{gx+hy+i}, \frac{dx+ey+f}{gx+hy+i})$$

Projective mappings are not preserved under 2D linear interpolation because the sum of two such expressions is in general a ratio of quadratics and therefore not a projective mapping. Consequently, morphing is a *shape-distorting* transformation, as in-between images may not correspond to new views of the same shape. A particularly disturbing effect of image morphing is its tendency to bend straight lines, yielding quite unintuitive image transitions. Fig. 3.5 shows a Dali-esque morph between two views of a clock in which it appears to bend in half and then straighten out again during the course of the transition. The in-between shapes were computed by linearly interpolating points in the two views that correspond to the same point on the clock.

## 3.7 Shape-Preserving Morphs: Parallel Views

We begin by considering situations in which linear interpolation of images is shape-preserving. Suppose we take a photograph $\mathcal{I}_0$ of an object, move the object in a direction parallel to the image plane of the camera, zoom out, and take a second picture $\mathcal{I}_1$. Alternatively, we could produce the same two images by moving the camera instead of the object, as shown in Fig. 3.6. Chen and Williams [CW93] previously considered this special case, arguing that linear image interpolation should produce new perspective views when the camera moves parallel to the image plane. Indeed, suppose that the camera is moved from the world origin to position $(C_X, C_Y, 0)$ and the focal length changes from $f_0$ to $f_1$. We write the respective projection matrices, $\mathbf{\Pi}_0$ and $\mathbf{\Pi}_1$, as:

$$\mathbf{\Pi}_0 = \begin{bmatrix} f_0 & 0 & 0 & 0 \\ 0 & f_0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

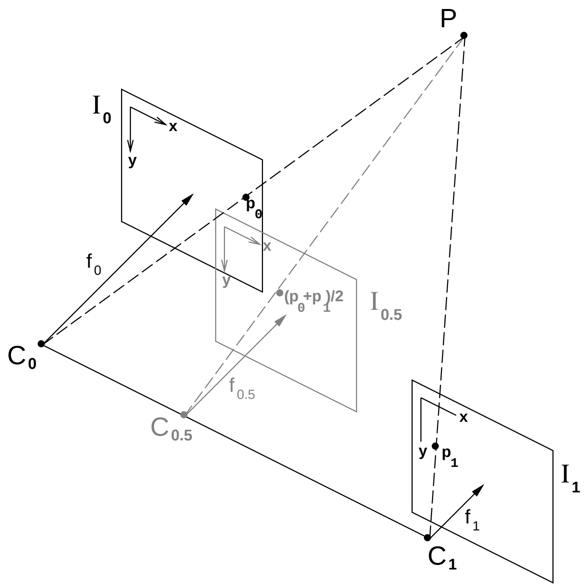


Figure 3.6: Morphing Parallel Views. Linear interpolation of corresponding pixels in parallel views with image planes $\mathcal{I}_0$ and $\mathcal{I}_1$ creates image $\mathcal{I}_{0.5}$, representing another parallel view of the same scene.

$$\mathbf{\Pi}_1 = \begin{bmatrix} f_1 & 0 & 0 & -f_1 C_X \\ 0 & f_1 & 0 & -f_1 C_Y \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

We refer to cameras or views with projection matrices in this form as *parallel cameras* or *parallel views*, respectively[1] Let $\mathbf{p}_0 \in \mathcal{I}_0$ and $\mathbf{p}_1 \in \mathcal{I}_1$ be projections of a scene point $\mathbf{P} = [X\ Y\ Z\ 1]^T$. Linear interpolation of $\mathbf{p}_0$ and $\mathbf{p}_1$ yields

$$\begin{aligned} (1-s)\mathbf{p}_0 + s\mathbf{p}_1 &= (1-s)\frac{1}{Z}\mathbf{\Pi}_0\mathbf{P} + s\frac{1}{Z}\mathbf{\Pi}_1\mathbf{P} \\ &= \frac{1}{Z}\mathbf{\Pi}_s\mathbf{P} \qquad (3.4) \end{aligned}$$

where

$$\begin{aligned} \mathbf{\Pi}_s &= (1-s)\mathbf{\Pi}_0 + s\mathbf{\Pi}_1 \\ &= \begin{bmatrix} f_s & 0 & 0 & -f_s\alpha_s C_X \\ 0 & f_s & 0 & -f_s\alpha_s C_Y \\ 0 & 0 & 1 & 0 \end{bmatrix} \\ f_s &= (1-s)f_0 + sf_1 \qquad (3.5) \\ \alpha_s &= \frac{sf_1}{(1-s)f_0 + sf_1} \qquad (3.6) \end{aligned}$$

[1] Not to be confused with *parallel projection* which produces *orthographic views*. Parallel views need not be orthographic.

Image interpolation therefore produces a new view whose projection matrix $\mathbf{\Pi}_s$ is a linear interpolation of $\mathbf{\Pi}_0$ and $\mathbf{\Pi}_1$, representing a camera with focal length $f_s$ and center $\mathbf{C}_s$ given by:

$$\mathbf{C}_s = (\alpha_s C_X, \alpha_s C_Y, 0) \tag{3.7}$$

Consequently, interpolating images from parallel cameras produces images that correspond to moving a camera on the line $\overline{\mathbf{C}_0\mathbf{C}_1}$ between the two optical centers and zooming continuously. Because the image interpolation produces new views of the same object, it is shape-preserving.

Notice that uniform parameterization of $s$ does not correspond to uniform camera motion, unless $f_0 = f_1$. Rather, a regular sampling of $s$ corresponds to a camera that moves more slowly as the focal length increases. For example, a halfway ($s = \frac{1}{2}$) image interpolation with $f_0 = 2$ and $f_1 = 1$ corresponds to a camera motion of only $\frac{1}{3}$ of the distance from $\mathbf{C}_0$ to $\mathbf{C}_1$, as seen in Fig. 3.6. In order to achieve constant velocity along $\overline{\mathbf{C}_0\mathbf{C}_1}$, i.e., uniform $\alpha$ parameterization, $s$ should be varied according to

$$s = \frac{\alpha f_0}{(1-\alpha)f_0 + \alpha f_1} \tag{3.8}$$

For example, in order to move the camera halfway from $\mathbf{C}_0$ to $\mathbf{C}_1$ in Fig. 3.6, we would substitute $\alpha = \frac{1}{2}$ into Eq. (3.8), yielding $s = \frac{2}{3}$.

The result that interpolations of parallel views is shape-preserving can be generalized; the above derivation in fact relies only on the equality of the third rows of $\mathbf{\Pi}_0$ and $\mathbf{\Pi}_1$. Views satisfying this more general criterion represent a broader class of parallel views for which linear image interpolation is shape preserving. An interesting special case is the class of orthographic projections, i.e., projections $\mathbf{\Pi}_0$ and $\mathbf{\Pi}_1$ whose last row is $[0\ 0\ 0\ 1]$. Linear interpolation of any two orthographic views of a scene therefore produces a new orthographic view of the same scene.

## 3.8 Occlusions and Visibility

So far, we have demonstrated that image morphing parallel views creates new images of a scene, in accordance with the equations of perspective projection. However, Eq. (2.1) does not model the effects that changes in *visibility* have on image content. In this section we examine both the case of constant visibility, in which unique in-between views are ensured, and changing visibility, in which *Z-buffer* techniques can be used to resolve ambiguities.

### 3.8.1 Constant Visibility

In order to validate image morphing as a technique for parallel view synthesis, we must relate the results of Section 3.7 to those of Section 3.5. In particular, it must be demonstrated that the synthesized images represent views in the same monotonic range. By doing so, we show that each image generated by morphing corresponds exactly to what a real camera would see from the same viewpoint.

Suppose that $\mathcal{I}_0$ and $\mathcal{I}_1$ are two monotonic parallel views with finite camera centers. Providing $s \in [0, 1]$, it follows from Eq. (3.7) that $\mathcal{I}_s$ is on the line between the camera centers and is therefore in the same monotonic range. Therefore, the proof of correctness is trivial for the case of finite camera centers.

For infinite camera centers, i.e., orthographic projection, the situation is more complicated. To preserve monotonicity, we must take care to ensure that the image plane normals $\mathbf{N}_s$ of the in-between views intersect the line segment $\overline{\mathbf{N}_0\mathbf{N}_1}$. Straightforward interpolation of orthographic views will generally not satisfy this property, as described in Appendix B. Fortunately, the situation can be easily corrected by adding the postwarp transformation of Section 3.9.2.

### 3.8.2 Changes in Visibility

From the standpoint of morphing, changes in visibility result in two types of conditions: *folds* and *holes*. A fold occurs in an in-between image $\mathcal{I}_s$ when a visible surface in $\mathcal{I}_0$ (or $\mathcal{I}_1$) becomes occluded in $\mathcal{I}_s$. In this situation, multiple pixels of $\mathcal{I}_0$ map to the same point in $\mathcal{I}_s$, causing an ambiguity. The opposite case, of an occluded surface suddenly becoming visible, gives rise to a hole; a region of $\mathcal{I}_s$ having no correspondence in $\mathcal{I}_0$.

Folds can be resolved using Z-buffer techniques [CW93], provided depth information is available. In the absence of 3D shape information, we use point *disparity* instead. The disparity of corresponding points $\mathbf{p}_0$ and $\mathbf{p}_1$ in two parallel views is defined to be the difference of their $x$-coordinates[2]. For parallel views, point disparity is inversely proportional to depth so that Z-buffer techniques may be directly applied, with inverse disparity substituted for depth. Because our technique makes images parallel prior to interpolation, as described in the next section, this simple strategy suffices in general. Furthermore, since the interpolation is computed one scanline at a time, Z-buffering may be performed at the scanline level, thereby avoiding the large memory requirements commonly associated with Z-buffering algorithms. An alternative method using a Painter's method instead of Z-buffering is presented in [MB95b].

Unlike folds, holes cannot always be eliminated using image information alone. Chen and Williams [CW93] suggested different methods for filling holes, using a designated background color, interpolation with neighboring pixels, or additional images for better surface coverage. The neighborhood interpolation approach is prevalent in existing image morphing methods and was used implicitly in our experiments.

[2]Here we assume that the images are rotated so that epipolar lines are horizontal and have the same top-to-bottom order in both images. Techniques for aligning images in this way are discussed in Section 3.9.

## 3.9 Shape-Preserving Morphs: Non-Parallel Views

In this section we describe how to generate a sequence of in-between views from two non-parallel perspective images of the same 3D object or scene. For convenience, we choose to model the transformation as a change in viewpoint, as opposed to a rotation and translation of the object or scene. The only tools used are image reprojection and linear interpolation, both of which may be performed using efficient scanline methods. For now, we assume the configuration of the input cameras is known; this constraint will be relaxed in Section 3.10.

### 3.9.1 Image Reprojection

Any two views that share the same optical center are related by a planar projective transformation. Let $\mathcal{I}$ and $\hat{\mathcal{I}}$ be two images with projection matrices $\mathbf{\Pi} = [\mathbf{H} \mid -\mathbf{HC}]$ and $\hat{\mathbf{\Pi}} = [\hat{\mathbf{H}} \mid -\hat{\mathbf{H}}\mathbf{C}]$. The projections $\tilde{\mathbf{p}} \in \mathcal{I}$ and $\tilde{\hat{\mathbf{p}}} \in \hat{\mathcal{I}}$ of any scene point $\mathbf{P}$ are related by the following transformation:

$$\begin{aligned} \hat{\mathbf{H}}\mathbf{H}^{-1}\tilde{\mathbf{p}} &= \hat{\mathbf{H}}\mathbf{H}^{-1}\mathbf{\Pi}\mathbf{P} \\ &= \hat{\mathbf{\Pi}}\mathbf{P} \\ &= \tilde{\hat{\mathbf{p}}} \end{aligned}$$

The $3 \times 3$ matrix $\hat{\mathbf{H}}\mathbf{H}^{-1}$ is a projective transformation that reprojects the image plane of $\mathcal{I}$ onto that of $\hat{\mathcal{I}}$. More generally, any invertible $3 \times 3$ matrix represents a planar projective transformation, a one-to-one map of the plane that transforms points to points and lines to lines. The operation of reprojection is very powerful because it allows the gaze direction to be modified *after* a photograph is taken or a scene rendered. Our use of projective transforms to compute reprojections takes advantage of an efficient scanline algorithm [Wol90]. Reprojection can also be performed through texture-mapping and can therefore exploit current graphics hardware.

Image reprojection has been used previously in a number of applications [Wol90]. Our use of reprojection is most closely related to the techniques used for rectifying stereo views to simplify 3D shape reconstruction [Fau93]. Image mosaic techniques [Gre86, MB95b, Che95, Sze96, SS97, KAI$^+$95, MP97] also rely heavily on reprojection methods to project images onto a planar, cylindrical, or spherical manifold. In the next section we describe how reprojection may be used to improve image morphs.

### 3.9.2 A Three Step Algorithm

Using reprojection, the problem of computing a shape-preserving morph from two non-parallel perspective views can be reduced to the case treated in Section 3.7. To this end, let $\mathcal{I}_0$ and $\mathcal{I}_1$ be two perspective views with projection matrices $\mathbf{\Pi}_0 = [\mathbf{H}_0 \mid -\mathbf{H}_0\mathbf{C}_0]$ and $\mathbf{\Pi}_1 = [\mathbf{H}_1 \mid -\mathbf{H}_1\mathbf{C}_1]$. It is convenient to choose the world coordinate system so that both $\mathbf{C}_0$ and $\mathbf{C}_1$ lie on the world $X$-axis, i.e., $\mathbf{C}_0 = [X_0\ 0\ 0]^T$ and $\mathbf{C}_1 = [X_1\ 0\ 0]^T$. The two remaining axes should be chosen in a way that

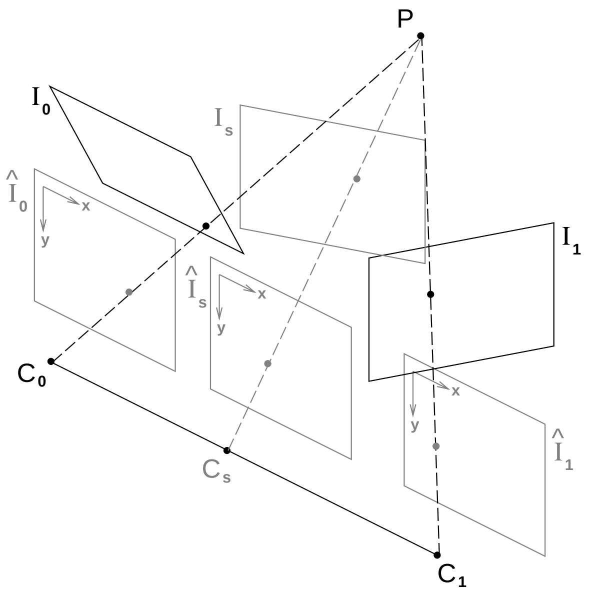


Figure 3.7: View Morphing in Three Steps. (1) Original images $\mathcal{I}_0$ and $\mathcal{I}_1$ are prewarped to form parallel views $\hat{\mathcal{I}}_0$ and $\hat{\mathcal{I}}_1$. (2) $\hat{\mathcal{I}}_s$ is produced by morphing (interpolating) the prewarped images. (3) $\hat{\mathcal{I}}_s$ is postwarped to form $\mathcal{I}_s$.

reduces the distortion incurred by image reprojection. Different choices for $Y$ and $Z$ correspond to different reprojection planes. A simple choice that works well in practice is to choose the $Y$ axis in the direction of the cross product of the two image plane normals.

In-between perspective views on the line $\overline{\mathbf{C}_0\mathbf{C}_1}$ may be synthesized by a combination of image reprojections and interpolations, depicted in Fig. 3.7. Given a projection matrix $\mathbf{\Pi}_s = [\mathbf{H}_s \mid -\mathbf{H}_s\mathbf{C}_s]$, with $\mathbf{C}_s$ fixed by Eq. (3.7), the following sequence of operations produces an image $\mathcal{I}_s$ corresponding to a view with projection matrix $\mathbf{\Pi}_s$:

1. **Prewarp:** Apply projective transforms $\mathbf{H}_0^{-1}$ to $\mathcal{I}_0$ and $\mathbf{H}_1^{-1}$ to $\mathcal{I}_1$, producing prewarped images $\hat{\mathcal{I}}_0$ and $\hat{\mathcal{I}}_1$
2. **Morph:** Form $\hat{\mathcal{I}}_s$ by linearly interpolating positions and colors of corresponding points in $\hat{\mathcal{I}}_0$ and $\hat{\mathcal{I}}_1$, using Eq. (3.4) or any image morphing technique that approximates it
3. **Postwarp:** Apply $\mathbf{H}_s$ to $\hat{\mathcal{I}}_s$, yielding image $\mathcal{I}_s$

Prewarping brings the image planes into alignment without changing the optical centers of the two cameras. Morphing the prewarped images moves the optical center to $\mathbf{C}_s$. Postwarping transforms the image plane of the new view to its desired position and orientation.

Notice that the prewarped images $\hat{\mathcal{I}}_0$ and $\hat{\mathcal{I}}_1$ represent views with projection matrices $\hat{\mathbf{\Pi}}_0 = [\mathbf{I} \mid -\mathbf{C}_0]$ and $\hat{\mathbf{\Pi}}_1 = [\mathbf{I} \mid -\mathbf{C}_1]$, where $\mathbf{I}$ is the $3 \times 3$ identity matrix. Due to the special form of these projection matrices, $\hat{\mathcal{I}}_0$ and $\hat{\mathcal{I}}_1$ have the property that corresponding points in the two images appear

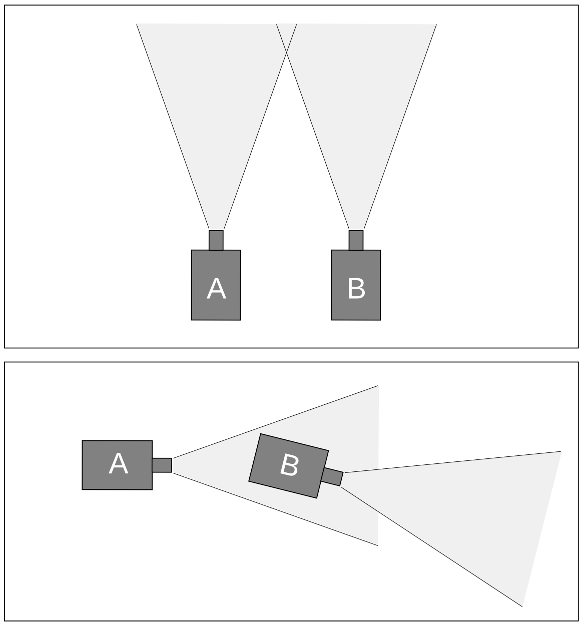


Figure 3.8: Singular Views. In the parallel configuration (top), each camera's optical center is out of the field of view of the other. A singular configuration (bottom) arises when the optical center of camera $B$ is in the field of view of camera $A$. Because prewarping does not change the field of view, singular views cannot be reprojected to form parallel views.

in the same scanline. Therefore, the interpolation $\hat{\mathcal{I}}_s$ may be computed one scanline at a time using only 1D warping and resampling operations.

### 3.9.3 Singular View Configurations

Certain configurations of views cannot be made parallel through reprojection operations. For parallel cameras, (Fig. 3.8, top) the optical center of neither camera is within the field of view of the other. Note that reprojection does not change a camera's field of view, only its viewing direction. Therefore any pair of views for which the optical center of one camera is within the field of view of the other cannot be made parallel through prewarping[3]. Fig. 3.8 (bottom) depicts such a pair of *singular* views, for which the prewarping procedure fails. Singular configurations arise when the camera motion is roughly parallel to the viewing direction, a condition detectable from the images themselves (see Appendix A). Singular views are not a problem when the prewarp, morph, and postwarp are composed into a single aggregate warp (see Section 3.11.2), since prewarped images are never explicitly constructed. With aggregate warps, view morphing may be applied to arbitrary pairs of views, including singular views.

[3]Prewarping is possible if the images are first cropped to exclude the epipoles (see Appendix A).

### 3.9.4 Producing the Morph

Producing a shape-preserving morph between two images requires choosing a sequence of projection matrices $\mathbf{\Pi}_s = [\mathbf{H}_s \mid -\mathbf{H}_s\mathbf{C}_s]$, beginning with $\mathbf{\Pi}_0$ and ending with $\mathbf{\Pi}_1$. Since $\mathbf{C}_s$ is determined by Eq. (3.7), this task reduces to choosing $\mathbf{H}_s$ for each value of $s \in (0, 1)$, specifying a continuous transformation of the image plane from the first view to the second.

There are many ways to specify this transformation. A natural one is to interpolate the orientations of the image planes by a single axis rotation. If the image plane normals are denoted by 3D unit vectors $\mathbf{N}_0$ and $\mathbf{N}_1$, the axis $\mathbf{D}$ and angle of rotation $\theta$ are given by

$$\begin{aligned} \mathbf{D} &= \mathbf{N}_0 \times \mathbf{N}_1 \\ \theta &= cos^{-1}(\mathbf{N}_0 \cdot \mathbf{N}_1) \end{aligned}$$

Alternatively, if the orientations are expressed using quaternions, the interpolation is computed by spherical linear interpolation [Sho85]. In either case, camera parameters such as focal length and aspect ratio should be interpolated separately.

## 3.10 Uncalibrated View Morphing

So far, we have assumed that the Euclidean camera positions for the two basis views and the synthesized view(s) are known. In this section we consider the case where the camera positions are not known and demonstrate that the view morphing algorithm may be extended to handle this *uncalibrated* case.

With today's technology, it is very difficult to obtain reliable estimates of camera positions, even in a controlled lab environment. Eliminating the need for camera calibration therefore makes the technique both easier to use and applicable in a wider range of situations. For instance, it becomes possible to apply the technique to drawings, paintings, and other images for which camera information is unavailable. In addition, the technique may be applied to interpolate views of *different objects* and to accommodate a range of 3D shape deformations.

### 3.10.1 Projective Transformations

By generalizing what we mean by a "view", the technique described in the previous section can be extended to accommodate a range of 3D shape deformations. In particular, view morphing can be used to interpolate between images of different 3D *projective* transformations of the same object, generating new images of the same object, projectively transformed. The advantage of using view morphing in this context is that salient features such as lines and conics are preserved during the course of the transformation from the first image to the second. In contrast, straightforward image morphing can cause severe geometric distortions, as seen in Fig. 3.5.

As described in Section 3.7, a 2D projective transformation may be expressed as a $3 \times 3$ homogeneous matrix transformation. Similarly, a 3D projective transformation is given by a $4 \times 4$ matrix $\mathbf{T}$. This class of transformations encompasses 3D rotations, translations, scales, shears, and tapering deformations. Applying $\mathbf{T}$ to a homogeneous scene point produces the point $\tilde{\mathbf{Q}} = \mathbf{TP}$. The corresponding point $\mathbf{Q}$ in 3D Euclidean coordinates is obtained by dividing $\tilde{\mathbf{Q}}$ by its fourth component. 3D projective transformations are notable in that they may be "absorbed" by the camera transformation. Specifically, consider rendering an image of a scene that has been transformed by a 3D projective transformation $\mathbf{T}$. If the projection matrix is given by $\mathbf{\Pi}$, a point $\mathbf{P}$ in the scene appears at position $\mathbf{p}$ in the image, where $\tilde{\mathbf{p}} = \mathbf{\Pi}(\mathbf{TP})$. If we define the $3 \times 4$ matrix $\tilde{\mathbf{\Pi}} = \mathbf{\Pi T}$, the combined transformation may be expressed as a single projection, representing a view with projection matrix $\tilde{\mathbf{\Pi}}$.

By allowing arbitrary $3 \times 4$ projections, we can model the changes in shape induced by projective transformations by changes in *viewpoint*. In doing so, the problem of interpolating images of projective transformations of an unknown shape is reduced to a form to which the three-step algorithm of Section 3.9.2 may be applied. However, recall that the three-step algorithm requires that the camera viewpoints be known. In order to morph between two different faces, this would require *a priori* knowledge of the 3D projective transformation that best relates them. Since this knowledge may be difficult to obtain, we describe here a modification that doesn't require knowing the projection matrices.

### 3.10.2 Computing Prewarps

Suppose we wish to smoothly interpolate two images $\mathcal{I}_0$ and $\mathcal{I}_1$ of objects related by a 3D projective transformation. Suppose further that only the images themselves and pixel correspondences are provided. In order to ensure that in-between images depict the same 3D shape (projectively transformed), $\mathcal{I}_0$ and $\mathcal{I}_1$ must first be transformed so as to represent parallel views. As explained in Section 3.9.2, the transformed images, $\hat{\mathcal{I}}_0$ and $\hat{\mathcal{I}}_1$, have the property that corresponding points appear in the same scanline of each image, i.e., two points $\mathbf{p}_0 \in \hat{\mathcal{I}}_0$ and $\mathbf{p}_1 \in \hat{\mathcal{I}}_1$ are projections of the same scene point only if their $y$-coordinates are equal. In fact, this condition is sufficient to ensure that two views are parallel, as shown in Appendix A. Consequently $\mathcal{I}_0$ and $\mathcal{I}_1$ may be made parallel by finding any pair of 2D projective transformations $\mathbf{H}_0^{-1}$ and $\mathbf{H}_1^{-1}$ that send corresponding points to the same scanline. One approach for determining $\mathbf{H}_0$ and $\mathbf{H}_1$ using 8 or more image point correspondences is given in Appendix A.

### 3.10.3 Specifying Postwarps

To fully determine a view morph, $\mathbf{H}_s$ must be provided for each in-between image. Rather than specifying the $3 \times 3$ matrix explicitly, it is convenient to provide $\mathbf{H}_s$ indirectly by establishing constraints on the in-between images. A simple yet powerful way of doing this is to interactively specify the paths of four image points through the entire morph transition. These control points can represent the positions of four point features, the endpoints of two lines, or the bounding quadrilateral of an arbitrary image

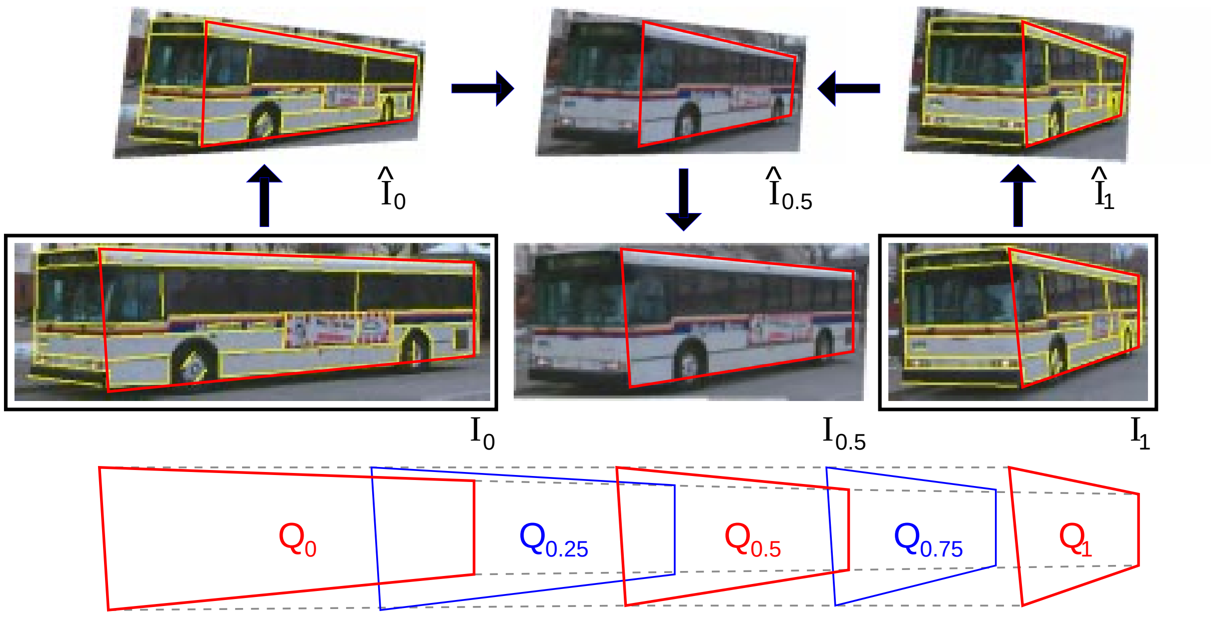


Figure 3.9: View Morphing Procedure: A set of features (yellow lines) is selected in original images $\mathcal{I}_0$ and $\mathcal{I}_1$. Using these features, the images are automatically prewarped to produce $\hat{\mathcal{I}}_0$ and $\hat{\mathcal{I}}_1$. The prewarped images are morphed to create a sequence of in-between images, the middle of which, $\hat{\mathcal{I}}_{0.5}$, is shown at top-center. $\hat{\mathcal{I}}_{0.5}$ is interactively postwarped by selecting a quadrilateral region (marked red) and specifying its desired configuration, $Q_{0.5}$, in $\mathcal{I}_{0.5}$. The postwarps for other in-between images are determined by interpolating the quadrilaterals (bottom).

region[4]. Fig. 3.9 illustrates the process: first, four control points bounding a quadrilateral region of $\hat{\mathcal{I}}_{0.5}$ are selected, determining corresponding quadrilaterals in $\mathcal{I}_0$ and $\mathcal{I}_1$. Second, the control points are interactively moved to their desired positions in $\mathcal{I}_{0.5}$, implicitly specifying the postwarp transformation and thus determining the entire image $\mathcal{I}_{0.5}$. The postwarps of other in-between images are then determined by interpolating the control points. The four curves traced out by the control points may also be manually edited for finer control of the interpolation parameters.

The positions of the control points in $\mathcal{I}_s$ and $\hat{\mathcal{I}}_s$ specify a linear system of equations whose solution yields $\mathbf{H}_s$ as follows. Each pair of control points $\mathbf{p} = [x\ y\ 1]^T$ and $\hat{\mathbf{p}} = [\hat{x}\ \hat{y}\ 1]^T$ are related according to:

$$c\hat{\mathbf{p}} = \mathbf{H}_s\mathbf{p} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \mathbf{p}$$

for an unknown scalar $c$. Eliminating $c$ yields two linear equations per control point in the coefficients of $\mathbf{H}_s$:

$$\begin{aligned} x(h_{31}\hat{x} + h_{32}\hat{y} + h_{33}) - h_{11}\hat{x} + h_{12}\hat{y} + h_{13} &= 0 \\ y(h_{31}\hat{x} + h_{32}\hat{y} + h_{33}) - h_{21}\hat{x} + h_{22}\hat{y} + h_{23} &= 0 \end{aligned}$$

Four control points yield eight such equations, which may be solved by adding the constraint $||\mathbf{H}_s|| = 1$ and using eigenvector techniques[5].

The use of image control points bears resemblance to the view synthesis work of Laveau and Faugeras [LF94], who used five pairs of corresponding image points to specify projection parameters. However, in their case, the points represented the projection of a new image plane and optical center and were specified only in the original images. In our approach, the control points are specified in the *in-between* image(s), providing more direct control over image appearance.

## 3.11 Implementation

This section discusses a number of issues that influence the quality and stability of images produced by view morphing. We first discuss methods for computing correspondence information and boundary flow from two basis images. Next, techniques for minimizing degradation from image resampling are considered. Finally, we note cases where the prewarp procedure becomes unstable and examine alternative solutions.

[4]Care should be taken to ensure that no three of the control points are colinear in any image.
[5]The solution of $\mathbf{Ah} = \mathbf{0}$, $||\mathbf{h}|| = \mathbf{1}$, is given by the unit eigenvector of $\mathbf{A}^T\mathbf{A}$ with eigenvalue 0.

### 3.11.1 Correspondence

The most difficult part of the view morphing procedure is determining the image correspondence information needed to compute the morph between prewarped images. We explored two different methods for computing correspondence, one based on stereo matching, and one that incorporates user interaction.

The first approach is to use stereo matching techniques to solve for point correspondences within epipolar lines. For our experiments, we used a dynamic programming method [OK85a] that exploits the monotonicity constraint. The prewarped images are in an ideal configuration for stereo matching because conjugate epipolar lines are horizontal and occupy corresponding scanlines in the two images. Therefore the images can be correlated one scanline at a time. Within each scanline, corresponding pixels are determined using the monotonicity constraint, which strongly limits the search space by fixing the relative pixel ordering. Edge segments are extracted to maintain inter-scanline constraints, as described in [OK85a].

As described in Section 3.5, a full pixel-correspondence is not needed to synthesize new views. Rather, we need only to determine boundary flow, i.e., correspondences of pixels at the boundaries of uniform intervals in epipolar lines. This modification is easily incorporated into existing scanline stereo matchers using the following preprocessing approach:

1. Segment each epipolar line $l$ into a sequence of intervals $i$, where the color of pixels in each interval is nearly uniform. In our implementation, this segmentation was performed by quantizing the RGB range of each image into a small number of colors (e.g., 30) and grouping consecutive pixels having the same color quantum into a single interval.

2. Construct a new line $l'$ consisting of a sequence of pixels, one for each interval in $i$. The color of the nth pixel in $l'$ corresponds to the average color of pixels in the nth interval of $i$.

3. For each pair of conjugate epipolar lines $l_0$ and $l_1$, match $l'_0$ and $l'_1$ using a scanline stereo matching technique, e.g., [OK85a]. The boundary flow is easily obtained from the resulting interval correspondence.

There are a number of improvements that could be made to this approach. To reduce the effects of salt and pepper noise, a higher weight could be assigned to pixels in $l'$ that correspond to large intervals of $l$. Note also that the segmentation technique of quantizing the image into $n$ colors for different values of $n$ could be used to generate a multi-scale representation over the set of uniform intervals. Specifically, note that small values of $n$ will cause epipolar lines to be segmented into a short sequence of large intervals, whereas large values of $n$ will favor numerous short intervals. Therefore, the stability of the approach could be improved by iterating the matching procedure for a sequence of increasing values of $n$.

A second approach to determining correspondence is to use user-interaction. Although this requires a human operator and may therefore be impractical for some applications, it allows for high quality interpolations of a much broader range of images and is robust to violations of the monotonicity assumption (for instance due to changes in lighting, variations in shape or color, or small-scale occlusions). For instance, it is possible to interpolate between views of *different objects* like human faces that have approximately the same shape. In the interactive approach, off-the-shelf image morphing programs can be used to compute the image correspondence and morph of the prewarped image. In one popular image morphing technique [BN92] a user interactively specifies a set of matching line segments in the two images. From these lines, the morph program interpolates a dense pixel correspondence. The endpoints of these same line segments are used to automatically prewarp the images, using the method described in Appendix A. The image warp is then computed by linearly interpolating the positions and colors of corresponding pixels in the two prewarped images.

The line-specification process is very intuitive and does not require advanced technical skills or artistic ability on the part of the user. In our experience, roughly 10 line features are often sufficient to create view morphs that give a good impression of the camera transformation, and 50-100 lines are sufficient to create high quality image transitions. The procedure is best performed incrementally: a user specifies a small number of lines, computes a view morph, examines the results, and adds more lines in image regions that need improvement.

### 3.11.2 Image Resampling

The prewarping and postwarping operations, combined with the intermediate morph, require multiple image resampling operations that may contribute to noticeable blurring in the in-between images. Resampling effects can be reduced by supersampling the input images [Wol90] or by composing the image transformations into one aggregate warp for each image. The latter approach is especially compatible with image morphing techniques that employ inverse mapping, such as the Beier and Neely method [BN92], since the inverse postwarp, morph, and prewarp can be directly concatenated into a single inverse map. Composing the warps has disadvantages however, including loss of both the scanline property and the ability to use off-the-shelf image morphing tools to compute the intermediate interpolation.

### 3.11.3 Prewarp Stability

The prewarp transformation, as described in Appendix A, relies on the *fundamental matrix* whose computation can be unstable. This instability is particularly prominent when feature positions are imprecise, or when the features lie on a variety of degenerate 3D configurations [LF96]. Poor prewarps are easily detected by the scanline property: corresponding points in two prewarped images should appear in the same scanline (i.e., have the same $y$ coordinate).

Note that in the application at hand, the features used to compute the prewarp are user-specified and may therefore be controlled to improve stability. For instance, in the presence of outliers it is best to compute the prewarp from a reliable subset of the image features. This is especially important when morphing between images of different objects that are approximately related by a 3D projective transformation. In this case, the user should point out the subset of features that appear to best fit a 3D projective transformation. Also, the user should be careful to avoid degenerate sets of feature correspondences, e.g., coplanar features. In addition, better results are obtained when the features are well-distributed throughout the pair of images. We have found that it is especially useful to include features near the image borders.

In some situations, however, the perspective prewarp procedure in Appendix A does not produce adequate results. For these cases, we instead propose a more stable *orthographic* prewarp procedure, which is also described in Appendix A. This technique is preferable for images that are nearly orthographic, including those taken with a telephoto lens and images of objects whose variation in depth is small relative to the distance from the camera. Because of the relatively high stability of the algorithm, we have found it to be a useful alternative for any pair of images in which the perspective prewarp algorithm fails.

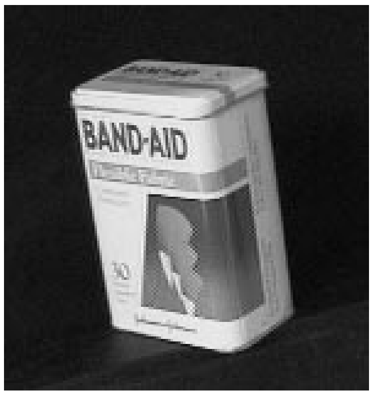
## 3.12 Experimental Results

The view morphing procedure was applied to several pairs of images to evaluate its performance. The first set of images, shown in Fig. 3.10, represent interpolations of views that were processed using stereo matching techniques [OK85a] to automatically derive boundary flow information. The images were prewarped by manually specifying 5-10 point correspondences and using the orthographic prewarp procedure described in Appendix A.

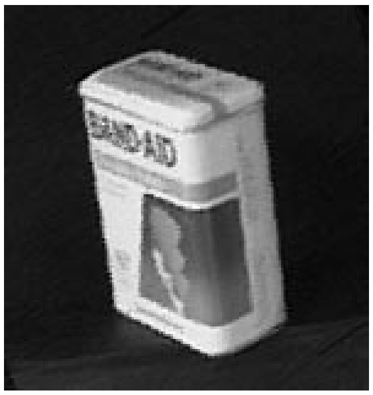
Fig. 3.10 shows three pairs of images and halfway interpolations for each pair. In each case, the postwarp was chosen by averaging the angles of rotation $\phi_0$ and $\phi_1$ and scale-translation matrix $\mathbf{T}$ of Appendix A.2.5. Notice that fine details such as the word "BAND-AID" are preserved in the interpolated images, despite the fact that each image is a product of several warps and resampling operations. The interpolated images were highpass filtered to help compensate for the blurring incurred by repeated image resampling.

The Band-Aid box (Fig. 3.10 (a-c)) is a good example of a scene for which monotonicity is satisfied. Images (d-f) and (g-i) pose a greater challenge becomes some regions that are visible in one image are occluded in the other. For example, a metallic surface of the stapler is visible in (d) but is completely occluded in (f) This region appears fuzzy in $\mathcal{I}_{0.5}$ due to the cross-dissolve between the two images. This type of artifact in image morphing is sometimes referred to as *ghosting*.

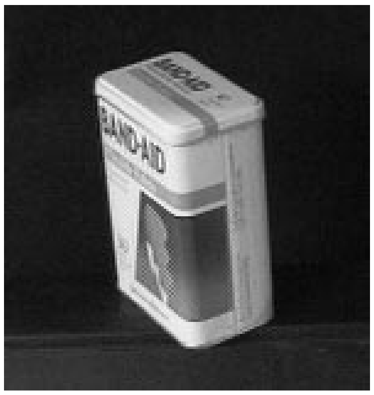
The final pair of images (g-i) contain a curved toy and a translucent salt shaker. Both objects violate monotonicity, the toy because of its self-occlusion at the contours, and the salt shaker because of its non-Lambertian surfaces. Still, the interpolation yields very realistic in-between images. We have found that local violations of the monotonicity assumption cause only local errors in corresponding regions of the interpolated images and do not corrupt the entire view morphing procedure. This property is reflected in image (e), where the occlusion of the stapler surface affected only a limited area in the interpolated image.



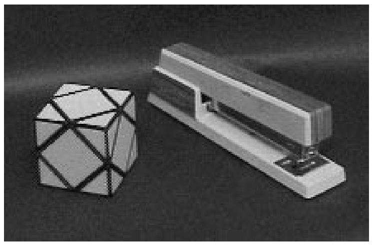
**(a)**



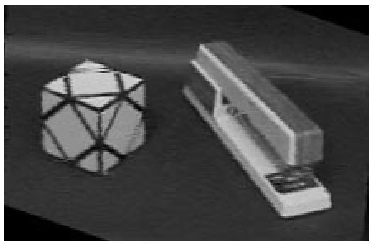
**(b)**



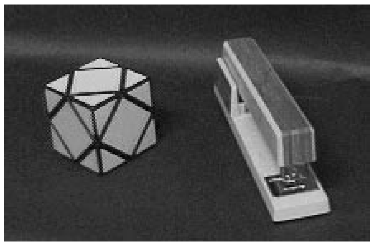
**(c)**



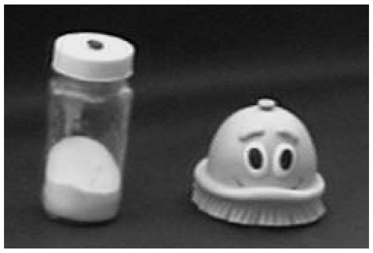
**(d)**



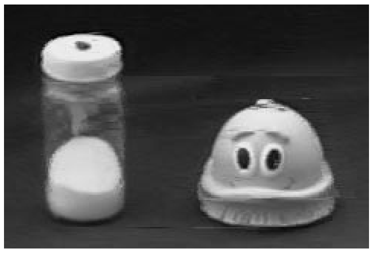
**(e)**



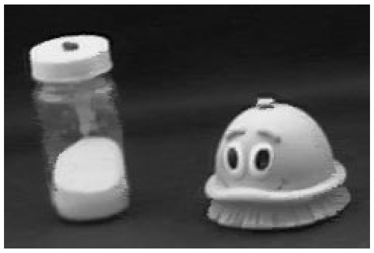
**(f)**



**(g)**



**(h)**



**(i)**

Figure 3.10: View morphs of three image pairs with automatic stereo matching. The original images $\mathcal{I}_0$ and $\mathcal{I}_1$ are shown in the left and right columns respectively, and an interpolated view $\mathcal{I}_{0.5}$ is shown at center.

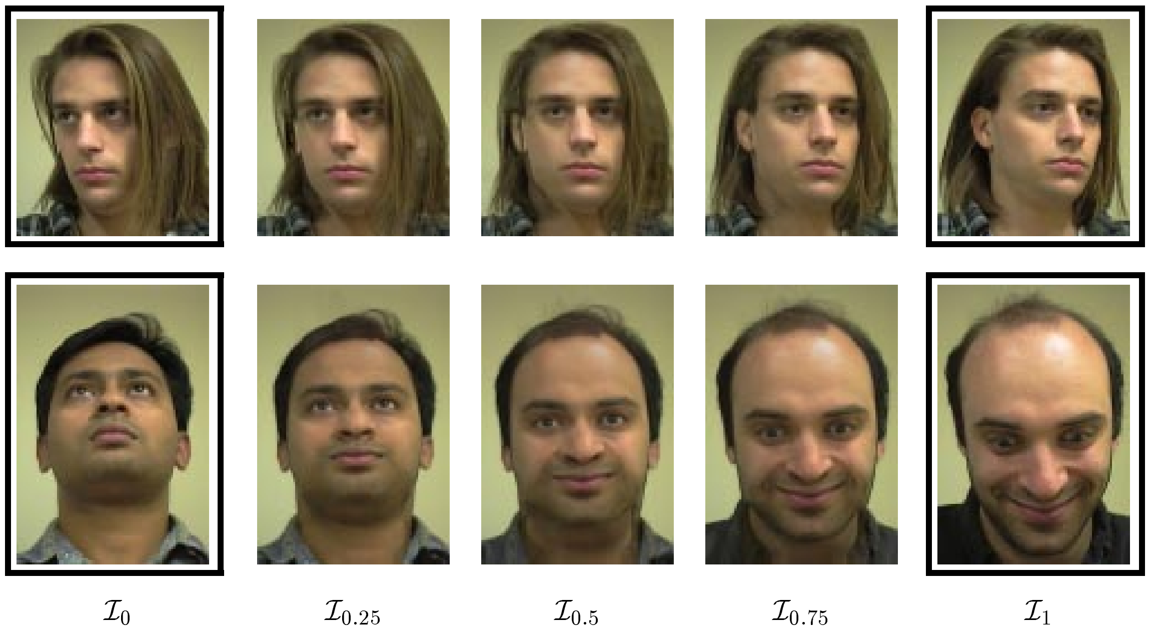


Figure 3.11: Facial View Morphs. Top: morph between two views of the same person. Bottom: morph between views of two different people. In each case, view morphing captures the change in facial pose between original images $\mathcal{I}_0$ and $\mathcal{I}_1$, conveying a natural 3D rotation.

The second set of images demonstrates the use of interactive techniques to facilitate the computation of image correspondences. For each pair of images in this set, we manually selected 50–100 corresponding line features. These features were used to automatically prewarp the images to achieve parallelism using the method described in Appendix A. Inspection of the prewarped images confirms that corresponding features, e.g., in Fig. 3.9, do in fact occupy the same scanlines. An implementation of the Beier and Neely field-morphing algorithm [BN92] was used to compute the intermediate images, based on the same set of features used to prewarp the images. The resulting images were postwarped by selecting a quadrilateral region delimited by four control points in $\hat{\mathcal{I}}_{0.5}$ and moving the control points to their desired positions in $\mathcal{I}_{0.5}$. The final positions of the control points for the image in the center of Fig. 3.9 were computed automatically by calibrating the two images based on their known focal lengths and interpolating the changes in orientation [Fau93].

Fig. 3.11 shows results for interpolating human faces in varying poses. The first example shows selected frames from a morph computed by interpolating views of the same person facing in two different directions. The resulting animation depicts the subject continuously turning his head from right to left. Because the subject's right ear is visible in only one of the original images, it appears "ghosted" in intermediate frames due to the interpolation of intensity values. In addition, the subject's nose appears slightly distorted as a result of similar changes in visibility. The second sequence shows a morph between different views of two *different faces*. Interpolating different faces is one of the most popular applications of image morphing. Here, we combine image morphing's capacity for dramatic facial interpolations with view morphing's ability to achieve changes in viewpoint. The result is a simultaneous interpolation of facial structure, color, and pose, giving rise to an image transition conveying a metamorphosis that appears strikingly 3D.

Fig. 3.12 compares image morphing with view morphing using two ray-traced images of a helicopter toy. The image morph was computed by linearly interpolating corresponding pixels in the two original images. The change in orientation between the original images caused the in-between images to contract. In addition, the bending effects seen in Fig. 3.5 are also present. Image morphing techniques such as [BN92] that preserve lines can reduce bending effects, but only when line features are present. An interesting side-effect is that a large hole appears in the image morph, between the stick and propeller, but not in the view morph, since the eye-level is constant throughout the transition. View morphs may also produce holes, but only as a result of a change in visibility.

Fig. 3.13 illustrates distortions that can occur in facial image morphs. In this example, Beier and Neely's method [BN92] was used to compute an image morph between an image of Leonardo da Vinci's *Mona Lisa* and its mirror reflection. The bending effects present in Fig. 3.12 are not as noticeable in this example, due to the smaller angle of rotation and the line-preserving properties of the Beir and Neely method. However, the contraction effects are strongly evident, especially when several in-between images are viewed in sequence. In particular, the face undergoes a horizontal contraction and appears unnaturally gaunt towards the half-way point of the image morph. Simultaneously, the neck and bust contract vertically, resulting in an accordion-like effect in which the figure compresses and then decompresses during the course of the image morph. In contrast, a view morph between the two images produces a more realistic transition that conveys a true 3D rotation of the head and torso.

## 3.13 Discussion

This chapter presented a detailed analysis of the view synthesis problem for the case of two input views. This problem is important because it provides a base case for more general view synthesis problems (Chapter 5 presents extensions to $n$ basis views). It is also interesting in its own right and has immediate applications of practical importance such as teleconferencing [TP94], face recognition [Sei97, SY97], stereo display [MB95a], and special effects [SD96c].

A main contribution was the uniqueness result for in-between views, which proved that all views of a scene between two cameras are uniquely determined from two basis images. Furthermore, these in-between views can be synthesized without 3D shape information, dense pixel correspondence, or knowledge of camera positions. A critical assumption was *monotonicity*, which dictates that no occlusions occur for the entire range of in-between views. The constraints of Lambertian surfaces and fixed illumination were also used to make the problem well-posed. This result is significant because it provides a theoretical basis for representing a scene's appearance with a collection of basis images. In particular, two views are sufficient to model appearance for the continuum of in-between views. In contrast, shape reconstruction is not generally possible from two views, due to ambiguities in the correspondence problem [Mar82, PTK85, VP89].

A second contribution was the *view morphing* algorithm for synthesizing high-quality in-between

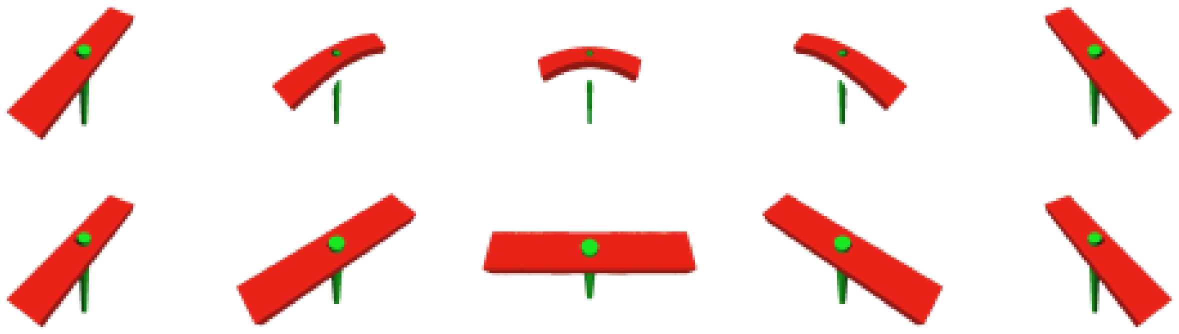


Figure 3.12: Image Morphing Versus View Morphing. Top: image morph between two views of a helicopter toy causes the in-between images to contract and bend. Bottom: view morph between the same two views results in a physically consistent morph. In this example the image morph also results in an extraneous hole between the blade and the stick. Holes can appear in view morphs as well.

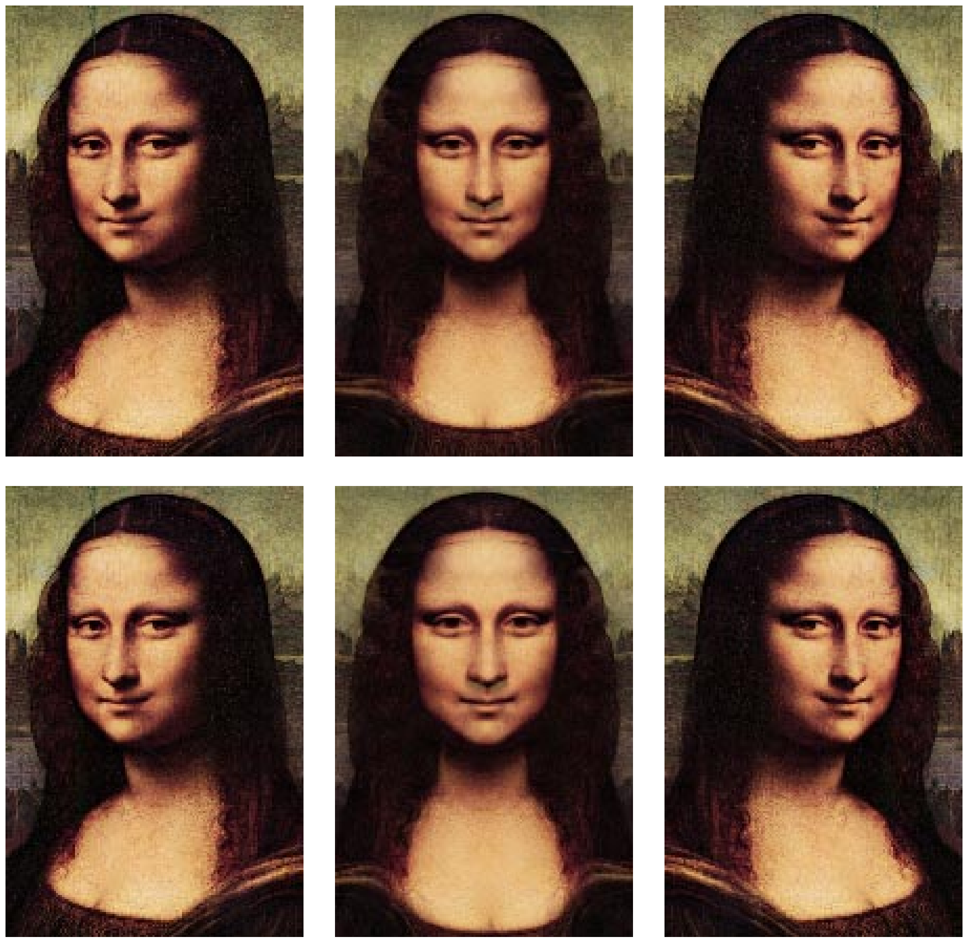


Figure 3.13: Top: image morph between Mona Lisa image and its reflection causes unnatural horizontal and vertical contraction of the face and torso. Bottom: view morph yields a more realistic transition that preserves the underlying 3D geometry.

views by warping two basis images. While the uniqueness result demonstrates that in-between views are theoretically determined, this algorithm provides a practical method for synthesizing high-quality in-between views from two basis images. Furthermore, whereas the theory requires strong assumptions, i.e., Lambertian surfaces, monotonicity, fixed illumination and shape, the algorithm demonstrates considerable robustness with respect to violations of these assumptions. Indeed, some of the most striking results were obtained for scenes in which the shading and shape differed significantly in the two basis images. In addition to a change in camera viewpoint, in these cases view morphing also conveys a realistic transformation of shape and illumination.

The best results were obtained when visibility was nearly constant, i.e., most scene points were visible in both basis views. Occlusion causes a *ghosting* effect, due to the cross-dissolve, in which unmatched points appear at fractional-intensity in in-between views. During the course of the morph transition, these points appear to slowly fade in or out. While barely noticeable in some sequences, ghosting becomes more of a problem when the unmatched region overlaps a perceptually salient feature in the scene. For instance, the right ear is occluded in one basis image image of Fig. 3.1 and appears unnatural in the in-between views.

The most challenging aspect of the view morphing approach is the correspondence problem, i.e., determining corresponding pixels or regions in the two basis images. Unlike most other view synthesis algorithms, view morphing does not require computing a complete optical flow map; rather, *boundary flow* is sufficient. While boundary flow is simpler to compute, it still requires solving a difficult stereo correspondence problem that is sensitive to image noise and quantization effects.

This chapter explored both automatic and manual solutions to the boundary correspondence problem. It is interesting to note that the “boundaries” in boundary flow typically correspond to perceptually salient features in the scene; in particular, surface and texture discontinuities represent the strongest boundaries. A consequence of this observation is that the performance of the view morphing algorithm is dependent most critically on the correspondence accuracy at these key features. Indeed, for the manual approach described in Section 3.11.1, a user typically specifies *only* these key features, e.g., eyes, nose, mouth, and silhouette. Correct correspondences at these features were sufficient to produce realistic morphs. In contrast, correspondence errors at these features yielded very poor morphs. The stereo correspondence techniques used in our implementation were much more likely to generate erroneous correspondences at these features than were human operators, and consequently tended to produce poorer results. Possible improvements could be obtained by using specialized feature detectors [BP93, PMS94, Per96] to facilitate automatic correspondence of key features like the eyes, nose, mouth. Another possibility would be to use a mixture of automatic and manual techniques, for instance using user-specified features to bootstrap a stereo refinement procedure, as in [DTM96].

# Chapter 4

# Single-View Morphing

When an object has bilateral symmetry, view morphs can be computed from a single image by interpolating the image with its mirror reflection. For example, Fig. 3.13 depicts a view morph between an image of Leonardo da Vinci's *Mona Lisa* and its reflection. Although the two sides of the face and torso are not perfectly symmetric, the morph conveys a convincing facial rotation.

The ability to artificially change the viewpoint of a single image suggests using view morphing as a means for *photo correction*. A portrait taken slightly off-center could be view morphed into a front-on view in the lab. Similarly, a professional photographer could interactively fine-tune an image's perspective after inspecting the developed photograph. These capabilities would add increased flexibility in the image acquisition process and provide the photographer with a powerful new set of image post-processing tools.

Photo correction also has potential uses in face recognition and image database applications. Most face recognition techniques [SI92, CWS95] use 2D pattern recognition methods to correlate an observed face with a database of reference images of known faces. In order to maintain low error rates, image acquisition must be carefully controlled to ensure that the facial pose is the same (e.g., frontal) in all views. To achieve better accuracy, these systems should compensate for differences in face orientation between the observed and reference images. One way of doing this is to normalize the images, using the procedure in Fig. 3.13, to convert both observed and reference images to front-facing views prior to recognition. An alternative approach that has recently been explored by several researchers [BP96, PMS94, MN90] is to estimate both pose *and* identity by acquiring images of each reference face from several different viewpoints. While this latter approach can cope with asymmetries and vertical head displacements, it is significantly more complicated and is only applicable when multiple views of each face are obtainable.

A few images from a hypothetical image database and their normalized versions are shown in Fig. 4.1. This figure also shows artifacts that can arise as a result of this normalization process. View morphing an image with its mirror reflection produces an image that is horizontally symmetric. Asymmetries in illumination or shape can produce unnatural-looking effects. The rightmost image in Fig. 4.1 is problematic because of two strong asymmetries—the illumination is one-sided and much of the right side of the head is occluded. As a result, the lighting and some facial contours in the normalized image appear distorted. Problems also occur consistently at hair lines, which appear unnaturally symmetric in the corrected images. These artifacts are relatively minor, however; the synthesized images clearly preserve the overall shape, color, and features of the original face and are easily

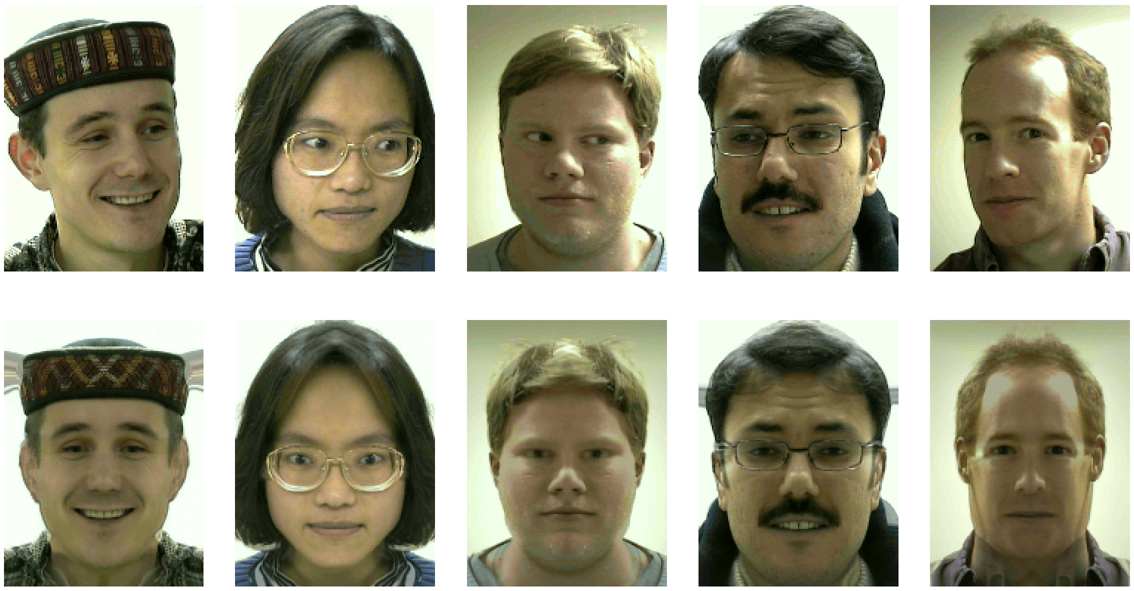


Figure 4.1: Photo correction. An image database of differently oriented faces (top) can be normalized using view morphing so that the faces are all horizontally front-facing (bottom).

recognizable to the human eye. Preliminary results by Seales and Yuan [SY97] indicate significant improvements in recognition accuracy when this approach is used to bootstrap 2D eigenface techniques [TP91].

# Chapter 5

# N-View Morphing

Chapter 3 focused on image synthesis from exactly two basis views. The two-view algorithm can in fact be used to compute interpolations between three or more views by reducing the problem to a series of two-view interpolations. The advantage of introducing additional basis images is that a greater range of views can be synthesized. Suppose for instance that we have three basis views that satisfy monotonicity pairwise (i.e., $(\mathcal{I}_0, \mathcal{I}_1)$, $(\mathcal{I}_0, \mathcal{I}_2)$, and $(\mathcal{I}_1, \mathcal{I}_2)$ each satisfy monotonicity). Three basis views permit synthesis of a triangular region of view space, delimited by the three optical centers. As shown in Fig. 5.1, each pair of basis images determines the views along one side of the triangle, spanned by $\overline{\mathbf{C}_0\mathbf{C}_1}$, $\overline{\mathbf{C}_1\mathbf{C}_2}$, and $\overline{\mathbf{C}_2\mathbf{C}_0}$.

## 5.1 Strong Monotonicity

What about interior views, i.e., views with optical centers in the interior of the triangle? Indeed, any interior view can be synthesized by a second interpolation, between a corner and a side view of the triangle. However, the assumption that monotonicity applies pairwise between corner views is not sufficient to infer monotonicity between views in the interior of the view triangle; monotonicity is not transitive. In order to predict interior views, a slightly stronger constraint is needed. Let $\triangle\mathbf{C}_0\mathbf{C}_1\mathbf{C}_2$ denote the closed triangular region delimited by $\mathbf{C}_0$, $\mathbf{C}_1$, and $\mathbf{C}_2$. *Strong monotonicity* dictates that for every pair of scene points $\mathbf{P}$ and $\mathbf{Q}$, the line $\mathbf{PQ}$ does not intersect $\triangle\mathbf{C}_0\mathbf{C}_1\mathbf{C}_2$. Strong monotonicity is a direct generalization of monotonicity; in particular, strong monotonicity of $\triangle\mathbf{C}_0\mathbf{C}_1\mathbf{C}_2$ implies that monotonicity is satisfied between every pair of views positioned in this triangle, and vice-versa. Consequently, strong monotonicity permits synthesis of any view in $\triangle\mathbf{C}_0\mathbf{C}_1\mathbf{C}_2$.

Now suppose we have $n$ basis views with optical centers $\mathbf{C}_0, \ldots, \mathbf{C}_{n-1}$ and strong monotonicity applies between each triple of basis views[1]. By the preceding argument, any triple of basis views determines the triangle of views between them. In particular, any view on the convex hull $\mathcal{H}$ of $\mathbf{C}_0, \ldots, \mathbf{C}_{n-1}$ is determined, as $\mathcal{H}$ is comprised of a subset of these triangles. Furthermore, the interior views are also determined: let $\mathbf{C}$ be a point in the interior of $\mathcal{H}$ and choose a corner $\mathbf{C}_i$ on $\mathcal{H}$. The line through $\mathbf{C}$ and $\mathbf{C}_i$ intersects $\mathcal{H}$ in a point $\mathbf{K}$. Since $\mathbf{K}$ lies on the convex hull, it represents the optical center of a set of views produced by two or fewer interpolations. Because $\mathbf{C}$ lies on $\overline{\mathbf{C}_i\mathbf{K}}$, any view positioned at $\mathbf{C}$ is determined as well by one additional interpolation, providing monotonicity is satisfied between $\mathbf{C}_i$ and $\mathbf{K}$. To establish this last condition, observe that for monotonicity to be

[1] In fact, strong monotonicity for each triangle on the convex hull of $\mathbf{C}_0, \ldots, \mathbf{C}_{n-1}$ is sufficient.

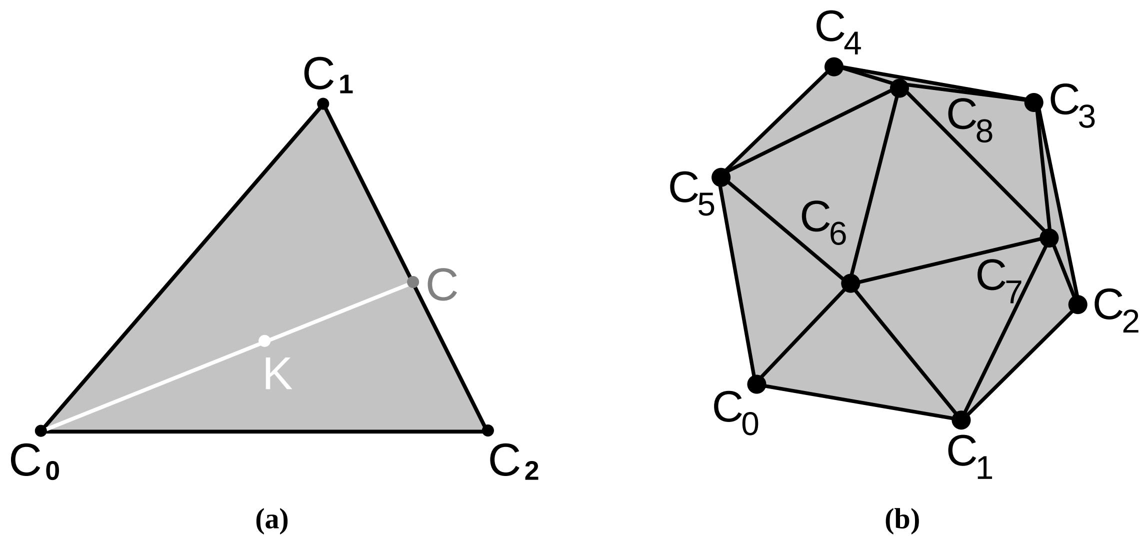


Figure 5.1: N-View Morphing. (a): The optical centers $\mathbf{C}_0$, $\mathbf{C}_1$, and $\mathbf{C}_3$ of three basis images $\mathcal{I}_0$, $\mathcal{I}_1$, and $\mathcal{I}_2$, determine a *view triangle*. If strong monotonicity applies, any view in the triangle may be synthesized. For example, a view $\mathcal{I}$ with optical center $\mathbf{C}$ is synthesized by interpolating $\mathcal{I}_1$ and $\mathcal{I}_2$. An interior view at $\mathbf{K}$ is produced by a subsequent interpolation of $\mathcal{I}$ and $\mathcal{I}_0$. (b): For $n$ basis images, any view within the convex hull of the camera centers can be generated with three interpolations.

violated there must exist two scene points $\mathbf{P}$ and $\mathbf{Q}$ such that $\mathbf{PQ}$ intersects $\overline{\mathbf{C}_i\mathbf{K}}$, implying that $\mathbf{PQ}$ also intersects $\mathcal{H}$. Thus, $\mathbf{PQ}$ intersects at least one triangle $\triangle\mathbf{C}_i\mathbf{C}_j\mathbf{C}_k$ on $\mathcal{H}$, violating the assumption of strong monotonicity. In conclusion, $n$ basis views determine the 3D range of viewspace contained in the convex hull of their optical centers.

## 5.2 Limitations

This constructive argument suggests that arbitrarily large regions of viewspace may be constructed by adding more basis views. However, the prediction of any range of view space depends on the assumption of strong monotonicity which dictates that no occlusions occur between any pair of views in the entire range. As noted in Section 3.5, a monotonic range may span no more than a single aspect of an aspect graph [SD96a], thus limiting the range of views that may be predicted.

In principle, a large range of views could be predicted by partitioning view space into monotonic ranges and using a different set of views to represent each range. This strategy is problematic, however, for moderately complex scenes and even some surprisingly simple objects. For instance, consider the case of a scene $\mathcal{S}$ containing a single Lambertian sphere. Any monotonic range of views of $\mathcal{S}$ contains at most one view, due to the fact that the set of visible points differs between any two views of a sphere. Therefore, a sphere represents a degenerate case for which view morphing cannot guarantee correct results. Although physically-correct results are not ensured, *realistic* results are often obtainable when monotonicity is violated. Fig. 3.10 (g-i) show an example similar to the sphere in which good results are obtained for a curved object.

This limitation of view morphing is due to the treatment of images as being atomic; each image is interpolated as a whole, thereby requiring constant visibility over the entire image. The other extreme would be to treat each pixel independently, as the projection of a point with its own visibility characteristics. This, in essence, is the approach taken in the next chapter, which provides a practical framework for view synthesis from numerous basis images.

# Chapter 6

# A Multi-View Approach: Voxel Coloring

View morphing demonstrates the feasibility of view synthesis and provides a robust algorithm for interpolating a pair of images. However, scene visibility is necessarily limited to what appears in only two basis views. In contrast, our goal in this chapter is to devise an algorithm capable of synthesizing *arbitrary* new views of a static scene from a set of basis views that are widely distributed about the environment. Specifically, our objectives are:

- **Photo-integrity**: The synthesized views should accurately reproduce the input images, preserving color, texture and pixel resolution
- **Broad Viewpoint Coverage**: new views should be accurate over a large range of target viewpoints. Therefore, the *basis views* should be widely distributed about the environment

In principle, adding more basis views should improve the fidelity of synthesized views. However, as noted in Chapter 5, the additional images introduce a whole range of new problems, like occlusion, calibration, correspondence, and representational issues. Whereas the two-image problem has been thoroughly studied in computer vision, theories of multi-image projective geometry, calibration, and correspondence have only recently begun to emerge [Sha94, Har94, LV94, Tri95, FM95, HA95]. Furthermore, the view synthesis problem, as presently formulated, raises a number of unique challenges that push the limits of existing multi-image techniques.

In this chapter, we describe an approach for view synthesis from multiple basis views that seeks to bypass the limitations of the two view approach, e.g., limited scene visibility, while retaining many of the theoretical and practical advantages of the view morphing algorithm, e.g., uniqueness properties and performance. Instead of approaching this problem as one of shape reconstruction, we formulate a *color reconstruction* problem, in which the goal is an assignment of colors (radiances) to points in an (unknown) approximately Lambertian scene. As a solution, we present a *voxel coloring* technique that traverses a discretized 3D space in "depth-order" to identify voxels that have a unique coloring, constant across all possible interpretations of the scene. This approach has several advantages over existing stereo and structure-from-motion approaches to pixel correspondence and scene reconstruction. First, occlusions are explicitly modeled and accounted for. Second, the cameras can be positioned far apart without degrading accuracy or run-time. Third, the technique integrates numerous images to yield dense reconstructions that are accurate over a wide range of target viewpoints.

The remainder of this chapter is structured as follows. Section 6.1 discusses the correspondence problem for view synthesis from numerous basis views and reviews capabilities and limitations of existing solutions. Section 6.2 introduces the voxel coloring paradigm as a means for determining correspondence and visibility in *scene space*. Section 6.3.1 defines the notion of *color invariants* and describes their importance for voxel coloring. Section 6.4 presents an efficient algorithm for computing a voxel coloring using a *layering* strategy, and Section 6.5 presents experimental results of applying the algorithm to real and synthetic basis images.

## 6.1 Pixel Correspondence for View Synthesis

In order to synthesize an image $\mathcal{I}_s$ of a scene from a set of basis images $\mathcal{I}_0, \ldots, \mathcal{I}_n$, we must determine the image color (irradiance) of each pixel $\mathbf{p} \in \mathcal{I}_s$. For Lambertian scenes, this problem reduces to that of computing the optical flow fields $C_{si} : \mathcal{I}_s \Rightarrow \mathcal{I}_i$ and using the relation

$$color(\mathbf{p}, \mathcal{I}_s) = color(C_{si}(\mathbf{p}), \mathcal{I}_i)$$

As noted in [PTK85, Bas92] however, optical flow computation is an inherently ill-posed problem and cannot be solved without additional assumptions on scene structure and camera placement. In Section 3.5 we used the assumption of monotonicity to enable a solution sufficient for view synthesis. However, to cope with changes in visibility, i.e., occlusions and changing field of view, a different approach is required.

In this section, we briefly review several approaches for computing pixel correspondences that have been used in view synthesis applications. Table 6.1 provides a qualitative comparison of these methods. The column headings represent conditions to which some approaches are sensitive, i.e., presence of such a condition imposes a performance penalty. Occlusion is caused by changes in scene visibility between basis views, and causes points to appear and disappear in different views. Large view separation is problematic for methods that assume small pixel displacements. Concave surfaces cannot be detected and therefore cause errors for techniques that operate solely on silhouette or occluding contour information. Note that Table 6.1 does not attempt to compare the relative *accuracy* of these different techniques. Rather, we seek to capture their relative strengths and weaknesses and identify the conditions in which each approach is most applicable. For comparison, we also list *voxel coloring*, the technique presented in this chapter.

### 6.1.1 Optical Flow

Avidan and Shashua [AS97] used optical flow correlation in conjunction with a trilinear tensor technique to generate new views from uncalibrated basis images. They achieved good results by correlating closely spaced frontal images of human faces. Faces present an ideal match for optical flow techniques due to the existence of individual views from which all points on the face are visible.

| Condition / Technique | occlusion | large view separation | surface concavity |
|---|---|---|---|
| optical flow [AS97] | | | $\surd$ |
| stereo [OLC93, KRN97, Sch96] | | | $\surd$ |
| feature tracking [TK92, BTZ96] | | | $\surd$ |
| contour tracking [CB92, SF95] | $\surd$ | | |
| volume intersection [MKKJ96, MTG97] | $\surd$ | $\surd$ | |
| EPI analysis [BBM87, KTOT95] | $\surd$ | | $\surd$ |
| Space-Sweep [Col96] | | $\surd$ | $\surd$ |
| voxel coloring [SD97b] | $\surd$ | $\surd$ | $\surd$ |

Table 6.1: Comparison of Correspondence Approaches. These approaches are evaluated with respect to their use for producing pixel correspondence maps under various conditions. A check indicates that a technique is *robust* with respect to a particular condition, i.e., the presence of that condition imposes virtually no performance penalty.

A variety of techniques exist for computing optical flow fields $C_{ij}$ (see [BFB94] for a survey and comparison). While it has been shown that optical flow estimates can be directly measured only at brightness discontinuities [Mar82, PTK85], smoothness conditions can be imposed to derive dense flow fields. Imposing smoothness is problematic, however, as it tends to produce poor results at motion discontinuities and occlusion boundaries. Furthermore, most optical flow techniques use gradient methods, window correlation, or velocity tuned filters, all of which require small image displacements.

We note that a number of techniques have been developed that are less susceptible to the aforementioned problems. For instance, *mixture-methods* and parameterized models have been proposed [WA94, DP95, BA96] to avoid smoothing over flow discontinuities. Other researchers have used coarse-to-fine schemes [BA83] to enable the recovery of flow fields with larger image displacements.

### 6.1.2 Stereo

Because of its relatively high accuracy, stereo is by far the most popular correspondence technique for view synthesis practitioners [OLC93, KRN97, WHH95, SD95b, MB95b, Sch96, DTM96]. However, problems with occlusions, widely separated views, and the need for specialized camera rigs make it impractical for many applications.

Stereo represents an adaptation of optical flow to the problem of matching two calibrated views of the same rigid scene. These additional constraints enable the problem to be reduced, in principle, to a series of one-dimensional searchs along conjugate epipolar lines. As such, stereo techniques tend to suffer from the same limitations as optical flow methods, namely problems with occlusions, surface discontinuity, and widely spaced views.

Recently, a number of researchers have devised techniques to improve the performance of binocular stereo in the presence of occlusion, e.g., using Bayesian techniques [BM92, GLY92] and masks [NMSO96] to detect half-occluded regions. Trinocular [SD88, AL91] and multiple-baseline stereo methods [OK85b] enable larger cumulative baselines and have been shown to improve matching accuracy. Still, individual views must generally be close together. Another approach for integrating disparate views is to arrange cameras in clusters and to compute stereo reconstructions from each cluster [KRN97]. The resulting partial models can be subsequently merged [TL94, CL96] to form a more complete reconstruction. This split and merge approach has proven most successful when the partial reconstructions to be merged are individually quite accurate, e.g., as obtainable by high-end laser range scanners. Integration of noisier stereo-derived models is significantly more difficult, if fine detail is to be preserved.

### 6.1.3 Feature Tracking

Several researchers have applied feature tracking techniques in association with shape-from-motion methods to recover high quality texture-mapped 3D models [TK92, BTZ96], suitable for view synthesis. However, because feature tracking does not yield dense correspondence fields, accuracy is ensured only in places where features have been detected and reliably tracked.

In contrast to optical flow and stereo algorithms whose flow fields are dense but often noisy, feature tracking methods seek to compute sparse but reliable correspondence maps. These latter methods detect correspondences only at fiducial features, e.g., corners or lines, and track the positions of these features throughout a dense sequence of images. In many cases, feature tracking approaches are able to reliably follow features over a long sequence of images, making this approach well-suited as an input source for shape-from-motion techniques. Problems occur, however, when features disappear (e.g., due to occlusion or detection failure), when the image displacement is large (e.g., due to large view separation), and when two similar features appear close together in an image. Solving the last problem requires combinatorial algorithms [Cox93] to keep track of all possible associations over a long image sequence. The search can be simplified by using heuristic pruning techniques [Cox93] or by imposing the assumption of a rigid scene [BCZ93, SD95a].

### 6.1.4 Contour Tracking and Volume Intersection

Moezzi et al. [MKKJ96, MTG97] applied a volume intersection technique to derive texture-mapped models for the purpose of view synthesis, using background subtraction to obtain silhouettes. Volume intersection techniques are attractive because, unlike most methods, they permit the input cameras to be arbitrarily far apart. However, they cannot detect concavities and are therefore less accurate than other methods.

Contour-based methods are a class of correspondence techniques that exploit the motion of the silhouette or occluding contours as a function of viewpoint to infer surface shape. These approaches can be characterized by whether or not they use temporal derivatives of the contour. While temporal derivatives provide useful shape and correspondence information [CB92, SF95], computation of temporal derivatives requires closely space views. Alternatively, surface shape can be inferred directly from the silhouette using *volume intersection* [MA91, Sze93] and other direct approaches [KD95a, Kut97].

A fundamental shortcoming of contour-based methods is that they fail at concavities and compute only the *visual hull* [Lau95, KD95b], an approximation of the true shape. Furthermore, they require the detection of occluding contours or silhouettes, a difficult problem in itself.

### 6.1.5 EPI Analysis

Motivated by applications in holography, Katayama et al. [KTOT95] adapted epipolar plane (EPI) analysis techniques for view synthesis. A key feature of their technique is that occlusions are explicitly modeled during the image correlation step. A constraint of the method is that the basis views must all lie along a line and be closely-spaced.

Epipolar plane analysis is a class of techniques introduced by Bolles, Baker, and Marimont [BBM87, BB89], to track and reconstruct features in closely-spaced views of a rigid scene. Stacking sequential images one on top of another forms an $XYT$ cube, called an *epipolar volume*. Under the constraint of linear camera motion, projected scene points trace out linear paths in slices of this epipolar volume. Therefore, point correspondences can be determined by applying line-detection techniques in individual EPI slices.

A key advantage of the EPI approach is its representation of occlusions. When two lines cross in the epipolar volume, the line with larger slope always corresponds to the occluder, i.e., the point which is closer to the axis of camera translation. Therefore, occlusions can be resolved by fitting lines in the order of decreasing slope [KTOT95].

### 6.1.6 Scene Space Methods

Determining correspondences using local image correlation methods, e.g., feature tracking or optical flow, is problematic when the motion between images is large. An alternative approach is to perform matching in *scene space* by detecting correspondences that are consistent with a rigid 3D scene. To our knowledge, scene space methods have not been used previously for view synthesis. However, they are similar in spirit to the method presented in this chapter and are consequently mentioned here as related work.

Seitz and Dyer [SD95a] proposed an approach that uses the correspondences of four point features in a sequence of views to determine the correspondences of all other features, using an assumption of scene rigidity. Each point or line feature in an image "votes" for the scene subspace that projects to that feature. Votes are accumulated when two or more subspaces intersect, indicating the possible presence of a point or line feature in the scene. A restriction of this technique is that it detects correspondences only for features that appear in all images.

Collins [Col96, Col97] proposed a *Space-Sweep* approach in which features are matched by sweeping a plane through the scene and accumulating votes for points on the plane that project to features in the images. Correspondences are identified by modeling the statistical likelihood of accidental accumulation and thresholding the votes to achieve a desired false positive rate. As with [SD95a], occlusions are not explicitly modeled in the Space-Sweep approach.

Zitnick and Webb [ZW96] described a scene space stereo technique that reconstructs scene regions that project unoccluded to the basis images. They noted that the correspondence problem is fundamentally ill-posed in the presence of occlusion, but can be solved when occlusion does not occur. By posing the problem as one of 3D surface extraction, it is possible to detect regions that are unoccluded in the basis images and thereby solve the correspondence problem for these image regions.

### 6.1.7 Discussion

Our objective in this chapter is to synthesize new views of a scene from a set of basis views that are widely distributed about the environment. Inspection of Table 6.1 indicates that existing techniques are not well-suited for providing the correspondence information sufficient for this purpose, due to problems with occlusion, camera separation, or concavities. In particular, these approaches do not guarantee *consistent* flow fields when occlusion is present, even under idealized conditions.

For instance, suppose we had two or more views $V_0, \ldots, V_n$ of a perfect Lambertian scene $\mathcal{S}$ under constant illumination, in which all internal and external camera parameters were precisely known. Furthermore, suppose all sources of error could be ignored, including those due to camera calibration, image quantization, and noise. While we could not hope to measure the flow-fields corresponding to the true scene [PTK85], we might reasonably expect to compute a *consistent* set of flow fields, i.e., one corresponding to some scene $\mathcal{S}'$ that appears identical to $\mathcal{S}$ from the basis viewpoints $V_0, \ldots, V_n$. However, existing algorithms do not guarantee a consistent set of flow fields when occlusion is present.

We believe that this shortcoming is due to the difficulty of reasoning about occlusion in image space, and instead advocate a scene space formulation for determining correspondence information. The advantages of this approach are two-fold: (1) it provides a framework for representing and analyzing the space of consistent scenes, and (2) the physical relations giving rise to occlusion are easily apparent and easily formalized. Therefore, scene space algorithms can be devised to take occlusions into account explicitly, and hence provide guarantees on the consistency of derived flow fields.

## 6.2 The Voxel Coloring Problem

In this section we introduce a new scene space framework for deriving dense correspondence information from multiple views $V_0, \ldots, V_n$. Rather than solve for the $n^2$ flow fields $C_{ij}$ directly, we instead reconstruct a colored, voxel-based 3D scene that is consistent with all of the basis views. This scene can either be reprojected to synthesize new views as in [SD97b], or used to compute correspondence maps for image-warping methods such as [CW93, LF94, MB95b, SD96c, Sch96]. The approach has the unique feature that it guarantees a consistent scene and set of flow fields when all assumptions are met.

The voxel coloring problem is to assign colors (radiances) to voxels (points) in a 3D volume so as to achieve consistency with a set of basis images, as shown in Fig. 6.1. That is, rendering the colored voxels from each basis viewpoint should reproduce the original image as closely as possible. More

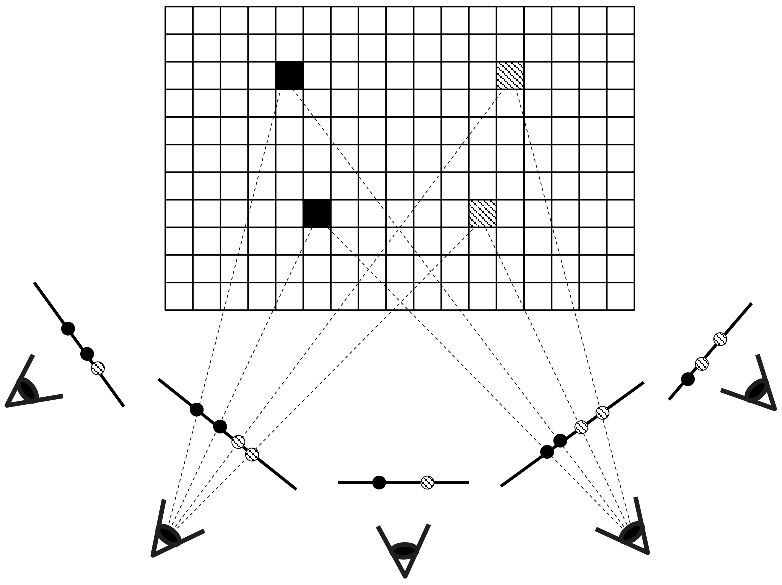


Figure 6.1: Voxel Coloring. Given a set of basis images and a grid of voxels, we wish to assign color values to voxels in a way that is consistent with all of the images.

formally, a 3D scene $\mathcal{S}$ is represented as a set of opaque Lambertian voxels (volume elements), each of which occupies a finite homogeneous scene volume centered at a point $\mathbf{V} \in \mathcal{S}$, and has a fixed color. We assume that the scene is entirely contained within a known, finite bounding volume. The set of all voxels in the bounding volume is referred to as the *voxel space* and denoted with the symbol $\mathcal{V}$. An image is specified by the set $\mathcal{I}$ of all its pixels, each centered at a point $\mathbf{p} \in \mathcal{I}$. For now, assume that pixels are infinitesimally small.

Given an image pixel $\mathbf{p} \in \mathcal{I}$ and scene $\mathcal{S}$, we refer to the voxel $\mathbf{V} \in \mathcal{S}$ that is visible in $\mathcal{I}$ and projects to $p$ by $\mathbf{V} = \mathcal{S}(\mathbf{p})$. A scene $\mathcal{S}$ is said to be *complete* with respect to a set of images if, for every image $\mathcal{I}$ and every pixel $\mathbf{p} \in \mathcal{I}$, there exists a voxel $\mathbf{V} \in \mathcal{S}$ such that $\mathbf{V} = \mathcal{S}(\mathbf{p})$. A complete scene is said to be *consistent* with a set of images if, for every image $\mathcal{I}$ and every pixel $\mathbf{p} \in \mathcal{I}$,

$$color(\mathbf{p}, \mathcal{I}) = color(\mathcal{S}(\mathbf{p}), \mathcal{S}) \tag{6.9}$$

We use the symbol $\aleph$ to denote the set of all consistent scenes. We may now define the voxel coloring problem formally:

> **Voxel Coloring Problem:** Given a set of basis images $\mathcal{I}_0, \ldots, \mathcal{I}_n$ and a voxel space $\mathcal{V}$, determine a subset $\mathcal{S} \subset \mathcal{V}$ and a coloring $color(\mathbf{V}, \mathcal{S})$, such that $\mathcal{S} \in \aleph$.

In order to solve this problem we must consider the following two issues:

- **Uniqueness**: Multiple voxel colorings may be consistent with a given set of images. How can the problem be well-defined?

- **Computation**: How can a voxel coloring be computed from a set of input images without combinatorial search?

Assuming that the images represent projections of the same Lambertian scene, it's clear that a consistent voxel coloring *exists*, corresponding to the set of points and colors on surfaces of the true scene[1]. Rarely, however, is the voxel coloring *unique*, given that a set of images can be consistent with more than one rigid scene. Determining a scene's spatial occupancy, i.e., $\mathcal{S}$, is therefore an ill-posed task because a voxel contained in one consistent scene may not be contained in another (see Fig. 6.2). Furthermore, a voxel may be contained in two consistent scenes, but have different colors in each (see Fig. 6.3). Consequently, additional constraints are needed in order to make the problem well-posed.

Computing voxel colorings poses another challenge. Observe that the underlying space is combinatorial: an $N \times N \times N$ grid of voxels, each with $M$ possible color assignments yields $2^{N^3}$ possible scenes and $M^{N^3}$ possible color assignments. Clearly, a bruce-force search through this space is not feasible.

## 6.3 Color Invariants

Given a multiplicity of solutions to the voxel coloring problem, the only way to recover intrinsic scene information is through *invariants*—properties that are satisfied by *every* consistent scene. For instance, consider the set of voxels that are contained in every consistent scene. Laurentini [Lau95] described how these invariants, called *hard points*, could be recovered by volume intersection from binary images. Hard points provide absolute information about the true scene but are relatively rare; some images may yield none (see, for example, Fig. 6.2). In this section we describe a more frequently occurring type of invariant relating to color rather than shape.

> A voxel $\mathbf{V}$ is a **color invariant** with respect to a set of images if: (1) $\mathbf{V}$ is contained in a scene consistent with the images, and (2) for every pair of consistent scenes $\mathcal{S}$ and $\mathcal{S}'$, $\mathbf{V} \in \mathcal{S} \cap \mathcal{S}'$ implies $color(\mathbf{V}, \mathcal{S}) = color(\mathbf{V}, \mathcal{S}')$.

Unlike shape invariance, color invariance does not require that a point be contained in every consistent scene. As a result, color invariants are more prevalent than hard points. In particular, it will be shown that the union of all color invariants itself yields a consistent scene, i.e., a complete voxel coloring, as depicted in Fig. 6.4. Therefore, the voxel coloring problem can be reformulated as a well-posed problem by solving for the consistent scene corresponding to the set of color invariants. In order to make the problem *tractable*, however, additional constraints are needed.

[1] This argument holds only in the limit, when voxels are infinitesimally small, or else when the true scene is *sampled*, i.e., representable as a finite collection of axis-aligned cubes.

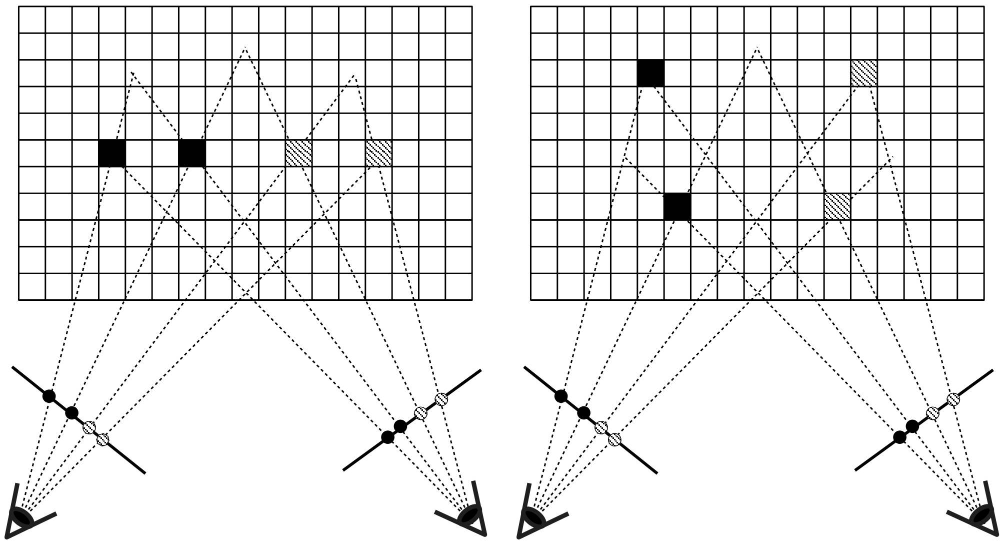


Figure 6.2: Spatial Ambiguity. Both voxel colorings appear identical from these two viewpoints, despite having no colored voxels in common.

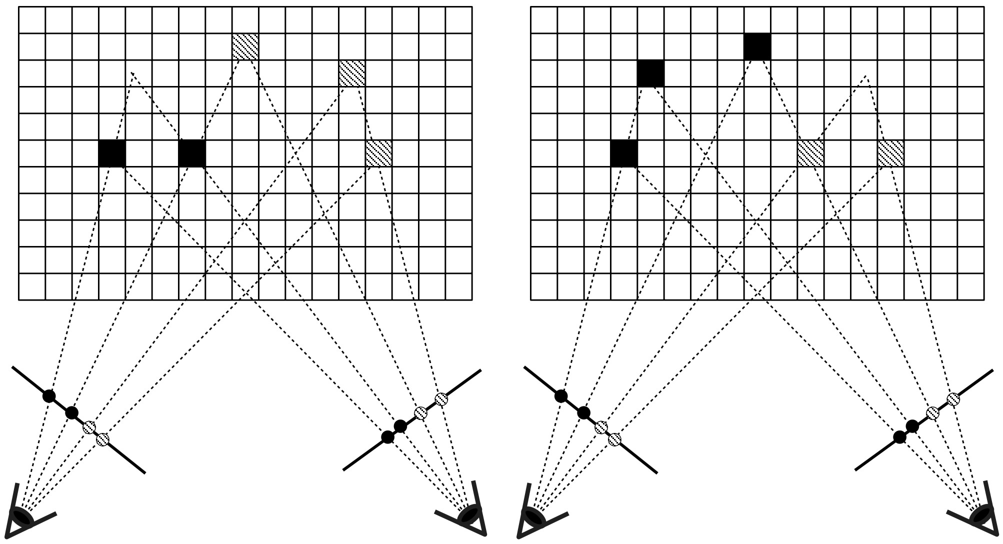


Figure 6.3: Color Ambiguity. Both voxel colorings appear identical from these two viewpoints. However, note the presence of a voxel (second row, center) that has a different color assignment in the two scenes.

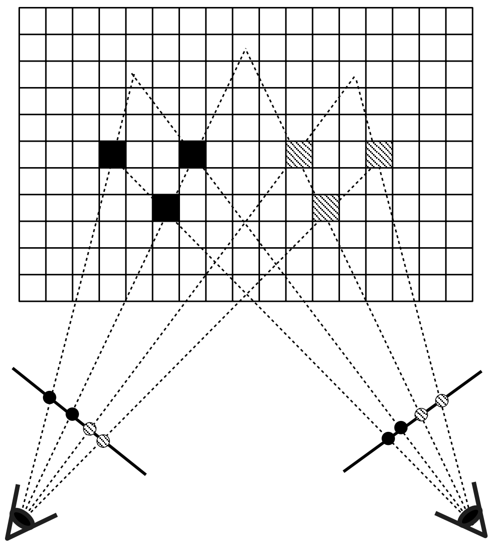


Figure 6.4: Color Invariants. Each of these six voxels has the same color in every consistent scene in which it is contained. The collection of all such *color invariants* forms a consistent voxel coloring denoted $\overline{\mathcal{S}}$, as depicted above. Note that the voxel with two color assignments in Fig. 6.3 is not contained in $\overline{\mathcal{S}}$.

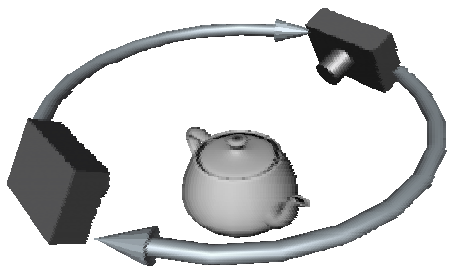
### 6.3.1 The Ordinal Visibility Constraint

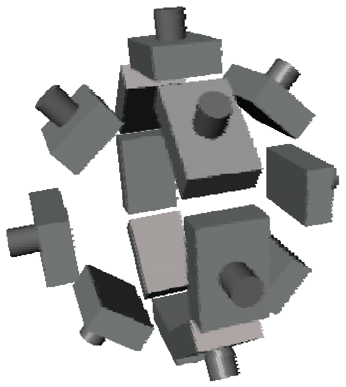
Note that color invariants are defined with respect to the set $\aleph$ of all consistent scenes—a combinatorial space. Clearly, an explicit search through this space is not computationally feasible. In order to make the problem tractable, we introduce a novel geometric constraint on camera placement relative to the scene that simplifies the analysis. This *ordinal visibility constraint* enables the identification of the set of color invariants as a limit point of $\aleph$. As such, they can be computed directly, via a single pass though the voxel space.

Let $\mathbf{P}$ and $\mathbf{Q}$ be scene points and $\mathcal{I}$ be an image from a camera centered at $\mathbf{C}$. We say $\mathbf{P}$ *occludes* $\mathbf{Q}$ if $\mathbf{P}$ lies on the line segment $\overline{\mathbf{CQ}}$. We require that the input cameras be positioned so as to satisfy the following constraint:

> **Ordinal visibility constraint:** There exists a norm $\|\cdot\|$ such that for all scene points $\mathbf{P}$ and $\mathbf{Q}$, and input images $\mathcal{I}$, $\mathbf{P}$ occludes $\mathbf{Q}$ in $\mathcal{I}$ only if $\|\mathbf{P}\| < \|\mathbf{Q}\|$.

We call such a norm *occlusion-compatible*. For some camera configurations, it is not possible to define an occlusion-compatible norm. However, a norm *does* exist for a broad range of practical configurations. For instance, suppose the cameras are distributed on a plane and the scene is entirely below that plane, as shown in Fig. 6.5(a). For every such viewpoint, the relative visibility of any two scene points depends entirely on which point is closer to the plane, so we may define $\|\cdot\|$ to be distance to the plane. More generally, the ordinal visibility constraint is satisfied whenever **no scene point is contained within the convex hull $\mathcal{C}$ of the camera centers**. Here we use the occlusion-compatible norm $\|\mathbf{P}\|_{\mathcal{C}}$, defined to be the Euclidean distance from $\mathbf{P}$ to $\mathcal{C}$. For convenience, $\mathcal{C}$ is referred to as the





**(a)** **(b)**

Figure 6.5: Compatible Camera Configurations. Both of the following camera configurations satisfy the ordinal visibility constraint: (a) an overhead inward-facing camera moved 360 degrees around an object, and (b) a rig of outward-facing cameras distributed around a sphere.

*camera volume*. Fig. 6.5 shows two useful camera configurations that satisfy this constraint. Fig. 6.5(a) depicts an inward-facing overhead camera rotating $360°$ around an object. Ordinal visibility is satisfied provided the camera is positioned slightly above the object. The constraint also enables "panoramic" configurations of outward-facing cameras, as in Fig. 6.5(b).

### 6.3.2 Properties of Color Invariants

To establish that color invariants exist, let $\mathcal{I}_0, \ldots, \mathcal{I}_n$ be a set of images for which the ordinal visibility constraint is satisfied. For a given image point $\mathbf{p} \in \mathcal{I}_j$ define $\mathbf{V_p}$ to be the voxel in $\{\mathcal{S}(\mathbf{p}) \mid \mathcal{S}\ consistent\}$ that is closest to the camera volume. We claim that $\mathbf{V_p}$ is a color invariant. To establish this, observe that $\mathbf{V_p} \in \mathcal{S}$ implies $\mathbf{V_p} = \mathcal{S}(\mathbf{p})$, for if $\mathbf{V_p} \neq \mathcal{S}(\mathbf{p})$, $\mathcal{S}(\mathbf{p})$ must be closer to the camera volume, a violation of our assumptions. It follows from Eq. (6.9) that $\mathbf{V_p}$ has the same color in every consistent scene, i.e., $\mathbf{V_p}$ is a color invariant.

Note that the preceding argument demonstrated not only that color invariants exist, but that *every* pixel in the basis images has a corresponding color invariant. We denote the collection of these color invariants as $\overline{\mathcal{S}}$:

$$\overline{\mathcal{S}} = \{\mathbf{V_p} \mid \mathbf{p} \in \mathcal{I}_i,\ 0 \leq i \leq n\}$$

It is easily shown that $\overline{\mathcal{S}}$ is a consistent scene. Note that $\overline{\mathcal{S}}$ is complete, since it contains a voxel corresponding to each pixel in the basis images. To show that it is consistent, for each $\mathbf{V} \in \overline{\mathcal{S}}$, choose $\mathbf{p} \in \mathcal{I}_i$, $0 \leq i \leq n$, such that $\mathbf{V} = \overline{\mathcal{S}}(\mathbf{p})$. Define

$$color(\mathbf{V}, \overline{\mathcal{S}}) := color(\mathbf{p}, \mathcal{I}_i) \tag{6.10}$$

To show that this coloring is well defined, suppose $\mathbf{p} \in \mathcal{I}_i$ and $\mathbf{q} \in \mathcal{I}_j$ are two points such that $\overline{\mathcal{S}}(\mathbf{p}) = \mathbf{V} = \overline{\mathcal{S}}(\mathbf{q})$. Let $\mathcal{S}$ be a consistent scene such that $\mathbf{V} \in \mathcal{S}$. By the definition of $\overline{\mathcal{S}}$, it follows that $\mathcal{S}(\mathbf{p}) = \mathbf{V} = \mathcal{S}(\mathbf{q})$. Hence, by Eq. (6.9),

$$color(\mathbf{p}, \mathcal{I}_i) = color(\mathbf{V}, \mathcal{S}) = color(\mathbf{q}, \mathcal{I}_j)$$

Therefore Eq. (6.10) is a well-defined voxel coloring and is consistent with the basis images.

Fig. 6.4 shows $\overline{\mathcal{S}}$ for the pair of images in Figs. 6.2 and 6.3. These six voxels have a unique color interpretation, constant in every consistent scene. They also comprise the closest consistent scene to the cameras in the following sense—every point in each consistent scene is either contained in $\overline{\mathcal{S}}$ or is occluded by points in $\overline{\mathcal{S}}$. An interesting consequence of this distance bias is that neighboring image pixels of the same color produce cusps in $\overline{\mathcal{S}}$, i.e., protrusions toward the camera volume. This phenomenon is clearly shown in Fig. 6.4, where the black and gray points form two separate cusps. Also, observe that $\overline{\mathcal{S}}$ is not a minimal reconstruction; removing the two closest points in Fig. 6.4 still leaves a consistent scene.

In summary, the following properties of color invariants have been shown:

- Every voxel in $\overline{\mathcal{S}}$ is a color invariant
- $\overline{\mathcal{S}} \subset \aleph$, i.e., it is a consistent scene
- $\overline{\mathcal{S}}$ represents a *limit point* of $\aleph$, corresponding to the consistent scene that is uniformly closest to the camera volume

## 6.4 A Voxel Coloring Algorithm

We now describe how to compute $\overline{\mathcal{S}}$ via a single pass through a discretized scene volume, by exploiting the ordinal visibility constraint. This constraint limits the possible basis view configurations, but the benefit is that visibility relationships are greatly simplified. In particular, it becomes possible to partition the scene into a series of voxel *layers* that obey a monotonic visibility relationship: for *every* input image, voxels only occlude other voxels that are in subsequent layers. Consequently, visibility relationships are resolved by traversing voxels one layer at a time.

### 6.4.1 Layered Scene Decomposition

To formalize this idea, we define the following partition of 3D space into voxel layers of uniform distance from the camera volume:

$$\mathcal{V}_d = \{V \mid \|V\| = d\} \tag{6.11}$$

$$\mathcal{V} = \bigcup_{i=1}^{r} \mathcal{V}_{d_i} \tag{6.12}$$

where $d_1, \ldots, d_r$ is an increasing sequence of numbers and $\| \cdot \|$ is an occlusion-compatible norm.

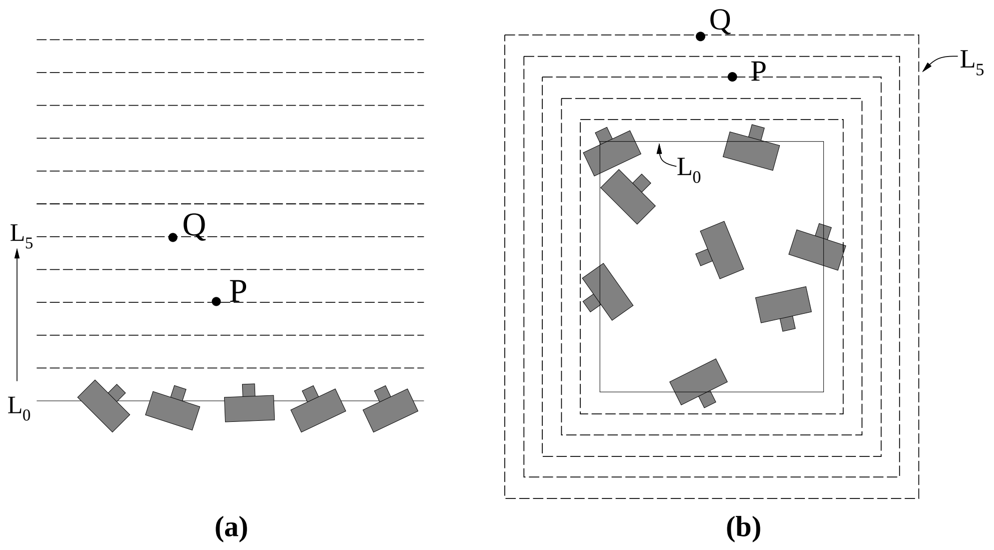


Figure 6.6: Layered Scene Traversal. Voxels can be partitioned into a series of layers of increasing distance from the camera volume. (a) Layers for cameras along a line. (b) Layers for cameras in a plane.

For the sake of illustration, consider a set of views positioned along a line facing a two-dimensional scene, as shown in Fig. 6.6 (a). Choosing $\|\cdot\|$ to be orthogonal distance to the line gives rise to a series of parallel linear layers that move away from the cameras. Notice that for any two voxels $\mathbf{P}$ and $\mathbf{Q}$, $\mathbf{P}$ can occlude $\mathbf{Q}$ from a basis viewpoint only if $\mathbf{Q}$ is in a higher layer than $\mathbf{P}$. The simplification of visibility relationships for this special case of colinear views was previously noted by Katayama et al. [KTOT95].

The linear case is easily generalized for any set of cameras satisfying the ordinal visibility constraint. Fig. 6.6 (b) shows a layer partition for the case of outward-facing cameras. This type of camera geometry is useful for acquiring *panoramic* scene visualizations, as in [MB95b, KS96]. One valid set of layers corresponds to a series of rectangles radiating outward from the camera volume. Layer $0$ is the axis-aligned bounding box $\mathcal{B}$ of the camera centers and the subsequent layers are determined by uniformly expanding the box one unit at a time. This set of layers corresponds to a norm given by the $L_\infty$ distance to $\mathcal{B}$.

Decomposing a 3D scene into layers can be done in the same manner. In the 3D case the layers become surfaces that expand outward from the camera volume. An especially useful layering strategy is the 3D analog of Fig. 6.6(b), in which each layer is an axis-aligned cube. The advantage of this choice of layers is that layers are computed and traversed very efficiently.

### 6.4.2 Voxel Consistency

To compensate for the effects of image quantization and noise, suppose now that the images are discretized on a grid of finite non-overlapping pixels. If a voxel $\mathbf{V}$ is not fully occluded in image $\mathcal{I}_j$, its projection overlaps a nonempty set of image pixels, $\pi_j$. Without noise or quantization effects, a consistent voxel should project to a set of pixels with equal color values. In the presence of these effects, we evaluate the correlation $\lambda_{\mathbf{V}}$ of the pixel colors to measure the likelihood of voxel consistency. Let $s$ be the standard deviation and $m$ the cardinality of $\bigcup_{j=0}^{n} \pi_j$. One possibility is to threshold the color space error:

$$\lambda_{\mathbf{V}} = s \tag{6.13}$$

Alternatively, a statistical measure of voxel consistency can be used. In particular, suppose the sensor error (accuracy of irradiance measurement) is normally distributed[2] with standard deviation $\sigma_0$. The consistency of a voxel can be estimated using the likelihood ratio test, distributed as $\chi^2$ with $n-1$ degrees of freedom [Fre92]:

$$\lambda_{\mathbf{V}} = \frac{(m-1)s^2}{\sigma_0^2} \tag{6.14}$$

If $\sigma_0$ is unknown, it can be estimated by imaging a homogeneous surface and computing the standard deviation $s_0$ of $m'$ image pixels. In this case, Eq. (6.14) should be replaced with

$$\lambda_{\mathbf{V}} = \frac{s^2}{s_0^2} \tag{6.15}$$

which has an $F$ distribution with $m-1$ and $m'-1$ degrees of freedom.

### 6.4.3 A Single-Pass Algorithm

In order to evaluate the consistency of a voxel, we must first compute $\pi_j$, the set of pixels that overlap $\mathbf{V}$'s projection in $\mathcal{I}_j$. Neglecting occlusions, it is straightforward to compute a voxel's image projection, based on voxel shape and the known camera configuration. We use the term *footprint*, following [Wes90] to denote this projection, corresponding to the intersection with the image plane of all rays from the camera center intersecting the voxel. Accounting for occlusions is more difficult, however, and we must take care to include only the images and pixel positions from which $\mathbf{V}$ should be *visible*. This difficulty is resolved by using the ordinal visibility constraint to visit voxels in an occlusion-compatible order and *marking* pixels as they are accounted for.

Initially, all pixels are unmarked. When a voxel is visited, $\pi_j$ is defined to be the set of *unmarked* pixels that overlap $\mathbf{V}$'s footprint in $\mathcal{I}_j$. When a voxel is evaluated and found to be consistent, all $m$ pixels in $\pi_j$ are marked. Because of the occlusion-compatible order of voxel evaluation, this strategy is sufficient to ensure that $\pi_j$ contains only the pixels from which each voxel is visible, i.e., $\overline{\mathcal{S}}(\mathbf{p}) = \mathbf{V}$ for each $\mathbf{p} \in \pi_j$. Note that by assumption voxels within a layer do not occlude each other. Therefore, the pixel marking phase can be delayed until after all the voxels in a layer are evaluated.

The complete voxel coloring algorithm can now be presented as follows:

[2]Here we make the simplifying assumption that $\sigma_0$ does not vary as a function of image intensity.

```
$\overline{\mathcal{S}} = \emptyset$
for $i = 1, \ldots, r$ do                                Iterate through the layers
   for every $\mathbf{V} \in \mathcal{V}_{d_i}$ do                   Iterate through voxels in the layer
      for $j = 0, \ldots, n$ do                          Project the voxel to each image
         compute footprint $\rho$ of $\mathbf{V}$ in $\mathcal{I}_j$
         $\pi_j = \{\mathbf{p} \in \rho \mid \mathbf{p}$ unmarked$\}$
      end for $j$
      compute $\lambda_{\mathbf{V}}$                            Evaluate voxel consistency
      if $m > 0$ and $\lambda_{\mathbf{V}} < thresh$ then
         $\overline{\mathcal{S}} = \overline{\mathcal{S}} \cup \{\mathbf{V}\}$                       Color the voxel
         $\pi = \pi \cup \bigcup_{j=0}^{n} \pi_j$                      Remember image pixels to mark
      end if
   end for $\mathbf{V}$
   mark pixels in $\pi$
end for
```

The threshold, $thresh$, corresponds to the maximum allowable correlation error. An overly conservative (small) value of $thresh$ results in an accurate but incomplete reconstruction. On the other hand, a large threshold yields a more complete reconstruction, but one that includes some erroneous voxels. Instead of thresholding correlation error, it is possible to optimize for model *completeness*. In particular, a completeness threshold $tcomp$ may be chosen that specifies the minimum allowable percentage of image pixels left unmarked. For instance, $tcomp = 75\%$ requires that at least three quarters of the (non-background) image pixels correspond to the projection of a colored voxel.

Given $tcomp$, we seek the minimum value of $thresh$ that yields a voxel coloring achieving this completeness threshold. Since completeness increases monotonically with $thresh$, it is sufficient to run the single-pass algorithm for a succession of increasing values of $thresh$, stopping when $tcomp$ is achieved. Alternatively, a binary search on $thresh$ may be used to decrease the number of iterations.

### 6.4.4 Discussion

The voxel coloring algorithm visits each of the $N^3$ voxels exactly once and projects it into every image. Therefore, the time complexity of voxel coloring is: $O(N^3 n)$. To determine the space complexity, observe that evaluating one voxel does not require access to or comparison with other voxels. Consequently, voxels need not be stored in main memory during the algorithm; the voxels making up the voxel coloring will simply be output one at a time. Only the images and one-bit mark masks need to be allocated. The fact that the space and time complexities of voxel coloring are linear in the number of images is essential so that large numbers of images can be processed at once.

The algorithm differs from stereo and tracking techniques in that it does not perform window-based image correlation during the reconstruction process. Correspondences are found during the course of scene traversal by voxel projection. A disadvantage of this searchless strategy is that it requires very precise camera calibration to achieve the triangulation accuracy of stereo methods. The effects of calibration and quantization errors are most significant at high-frequency regions such as image edges. Preserving high-frequency image content requires a higher voxel sampling rate because of Nyquist considerations. However, smaller voxels result in fewer pixels being integrated in the correlation step and therefore are more sensitive to calibration errors. An extension would be to compensate for high-frequency regions in the correlation step, for instance by detecting and treating edges specially.

Accuracy and run-time also depend on the voxel resolution, a parameter that can be set by the user or determined automatically to match the pixel resolution, calibration accuracy, and computational resources. An extension would be to use *hierarchical* representations like octrees [Sam84, Sze93] in which the voxel resolution is locally adapted to match surface complexity. Maintaining the strict voxel visitation order within a hierarchical framework may be difficult, however, and is beyond the scope of this thesis.

Importantly, the voxel coloring approach reconstructs only one of the potentially numerous scenes consistent with the input images. Consequently, it is susceptible to aperture problems caused by image regions of near-uniform color. These regions cause cusps in the reconstruction (see Fig. 6.4), since voxel coloring yields the reconstruction closest to the camera volume. This is a bias, just like smoothness is a bias in stereo methods, but one that guarantees a consistent reconstruction even with severe occlusions.

### 6.4.5 Optimizations

Much of the work of the algorithm lies in the computations of $\pi_j$ and $\lambda_{\mathbf{V}}$. For simplicity, our implementation used a square mask to approximate voxel footprints, and used Eq. (6.13) to test voxel consistency. Alternative footprint models are discussed in the volume rendering literature, e.g., [Wes90, LH91].

While our implementation did not make use of this, additional speedups are possible by exploiting the uniform discretization of space and simple layer geometry. Choosing planar or polyhedral layers enables the use of texture-mapping graphics hardware to calculate voxel footprints, an entire layer at a time. This strategy enables more accurate estimates of voxel footprints and offloads most of the computation to the graphics co-processor. For instance, the projection by $\mathbf{\Pi}_i$ of a plane layer

$$\mathbf{V}_{u,v} = \mathbf{V}_{0,0} + u\mathbf{D}_X + v\mathbf{D}_Y$$

can be expressed in matrix form by $\mathbf{\Pi}_i\mathbf{V}_{u,v} = \mathbf{H}_i[u\ v\ 1]^T$, where the $3 \times 3$ homography $\mathbf{H}_i$ is given by

$$\mathbf{H} = [\mathbf{\Pi}_i\mathbf{D}_X \mid \mathbf{\Pi}_i\mathbf{D}_Y \mid \mathbf{\Pi}_i\mathbf{V}_{0,0}]$$

Instead of projecting the layer onto each image, it is preferable to reverse-map each image onto the layer by applying

$$\hat{\mathcal{I}}_i = \mathbf{H}_i^{-1}\mathcal{I}_i$$

This procedure allows the voxel projections to be directly integrated; the footprint of voxel $\mathbf{V}_{u,v}$ in $\mathcal{I}_i$

is simply the pixel at position $[u\ v\ 1]^T \in \hat{\mathcal{I}}_i$. This reverse-mapping strategy is similar to that of Collins [Col96], who used a diffusion operator in place of the texture-mapping formulation.

## 6.5 Experimental Results

In order to evaluate the performance of the voxel coloring algorithm for view synthesis, it was applied to images of a variety of real scenes. Synthetic images were also used to facilitate analysis of error characteristics, and to simulate camera configurations that were not physically realizable in the lab.

### 6.5.1 Results on Real Images

The first experiment demonstrates the view synthesis capabilities of the voxel coloring algorithm, as applied to images of real objects. Calibrated images were captured with the aid of a computer-controlled pan-tilt head and a fixed overhead camera, as shown in Fig. 6.8. This strategy is similar to that in [Sze93]. An object was placed on the head and rotated 360 degrees in front of a color camera (Sony XC-999 with 1/2" CCD sensor and 12mm, F1.4 lens) positioned approximately 30cm horizontally from the object's center and 25cm vertically above it's base. Tsai's method [Tsa87] was used to calibrate the camera with respect to the head, by rotating a known object and manually selecting image features for three pan positions. The calibration error was approximately 3%. Fig. 6.7 shows selected images for two objects: a toy dinosaur (6cm x 8cm x 10cm) and a rose. In each case 21 input images ($640 \times 486$ resolution) were captured by rotating the object 360 degrees in increments of about 17 degrees. A problem with this acquisition approach is that the illumination effectively changes as the object rotations, thereby violating the Lambertian assumption. In compensation, the error threshold was set relatively high: 18% pixel correlation error was allowed for the dinosaur, and for 12% for the rose.

Table 6.2 compares sizes, run times, and reprojection errors for voxel colorings computed from the dinosaur toy at four different resolutions. Square voxels were used and the grid volume was held constant. The resolution was specified by the grid dimensions, which indicate the total number of voxels in the volume (width x depth x height). Each row in Table 6.2 represents a resolution doubling, i.e., an 8-fold increase in the number of voxels, relative to the previous row. The run time increases proportionately, although not quite by the factor of 8. This is attributed to an additive overhead factor of the algorithm. These run times do not include image acquisition, calibration, or thresholding. The "Voxels Colored" column indicates the number of voxels selected and output by the voxel coloring algorithm. This is the effective size of the reconstruction. Notice that the voxels colored column increases more slowly than the voxels evaluated column. A probable reason is that the voxel coloring algorithm reconstructs only points on the *surface*, i.e., not in the interior of the object. It would therefore be expected that the voxels colored would increase only by a factor of 4 when the resolution is doubled, at least for 2D manifold surfaces. The final column of Table 6.2 gives the reprojection error, which measures the root-mean-squared error of pixels in synthesized images for each of the

21 input viewpoints. Clearly, increasing the voxel resolution has a significant impact on reprojection error.

Fig. 6.9 shows the voxel colorings of the dinosaur toy described in Table 6.2. To facilitate reconstruction, a black background was used and the images were thresholded to eliminate most of the background points. While background segmentation is not strictly necessary, leaving this step out results in background-colored voxels scattered around the edges of the scene volume. The threshold may be chosen conservatively since removing most of the background pixels is sufficient to eliminate this background scattering effect. Fig. 6.9(c) shows the highest resolution reconstruction from a viewpoint corresponding to one of the input image. For comparison, the original image is also shown in Fig. 6.9(a). Note that even fine details such as the wind-up rod on the dinosaur were reconstructed, as seen more clearly in (g).

Fig. 6.9(d-g) compare reconstructions from a new viewpoint, different than the basis views, for the different voxel resolutions reported in Table 6.2. The resolution doubles at each step from (d-g). Note that even the lowest resolution model, shown in (d), preserves the rough features of the model and produces a very reasonable reprojection. The fact that this model was computed in 3 seconds suggests that the algorithm is potentially suitable for interactive applications like teleconferencing, in which models must be generated in real time. Increasing the resolution adds fine details in shape and texture and decreases the blocky artifacts caused by large voxels, as seen in (e-g).

Results for the rose are shown in Fig. 6.10. (a) shows an input image and (b), a synthesized view for the same viewpoint, demonstrating the photo-integrity of the reconstruction. The rose represents a difficult case because it has little texture and is made up of surfaces like leaves and petals that are extremely thin. The reconstruction, consisting of approximately 70,000 voxels, captures the appearance of the rose very accurately, and preserves most of these fine features. (c) and (d) show synthesized views from new views that are close to and far away from the basis views, respectively. Overall, the image in (d) is quite good, but it exhibits some interesting artifacts. Specifically, the leaves appear to contain holes when viewed from below. These holes are not visible from the basis viewpoints and so were not filled in by the algorithm—they represent unreconstructible regions in the scene.

### 6.5.2 Results on Synthetic Images

In order to evaluate the performance of the voxel coloring algorithm for *panoramic* scenes, basis images were generated by placing several cameras in a synthetic room scene. The room consisted of three texture-mapped walls and two shaded figures. The figures, a bust of Beethoven and a scanned Cyberware model of a human figure, were illuminated diffusely from a downward-oriented light source at infinity. 24 cameras were placed at different positions and orientations *inside* the room, as shown in Fig. 6.11.

The geometry of this scene and the camera configuration would pose significant problems for previous image-based reconstruction methods. In particular, the room interior is highly concave, making accurate reconstruction by volume intersection or other contour-based methods impractical. Furthermore, the numerous cameras and large amount of occlusion would create difficulty for most stereo approaches. Notable exceptions include panorama-based stereo approaches [MB95b, KS96] that are well-suited for room reconstructions. However, these methods require that a panoramic image be constructed for each camera location prior to the stereo matching step, a requirement that is avoided by the voxel coloring approach. This requirement does not enable camera configurations such as the one shown in Fig. 6.11.

Fig. 6.12 compares the original and reconstructed models of the room from new viewpoints. The reconstruction contained 320,000 voxels and required 45 minutes to compute. The voxel coloring reproduced images from the room interior extremely accurately, as shown in (b). A pixel correlation error threshold of 2.4% was used to account for image quantization. As a result of these errors, some fine details were lost, e.g., in the face of the Beethoven bust. The overhead views (d) and (f) more clearly show some discrepancies between the original and reconstructed models. For instance, the reconstructed walls are not perfectly planar, as some points lie just off the surface. This point drift effect is most noticeable in regions where the texture is locally homogeneous, indicating that texture information is important for accurate reconstruction. The quality of the overhead view shown in (d) is especially commendable, given that the viewpoint is very far away from the input views. The extreme overhead view (f) is worse than that of (d) but clearly shows that the overall shape of the scene was very well captured by reconstruction.

A second set of experiments was conducted to evaluate the sensitivity of the approach to factors of texture density, image noise, and voxel resolution. To simplify the analysis of these effects, the experiments were performed using a 2D implementation of the voxel coloring method for which the scene and cameras lie in a common plane. Fig. 6.13(a) shows the synthetic scene (an arc) and the positions of the basis views used in these experiments.

Texture is an important visual cue, and one that is exploited by voxel coloring. To model the influence of texture on reconstruction accuracy, a series of reconstructions were generated in which the texture was systematically varied. The spatial structure of the scene was held fixed. The texture pattern was a cyclic linear gradient, specified as a function of frequency $\theta$ and position $t \in [0, 1]$:

$$intensity(t) = 1 - |1 - 2 * frac(\theta * t)|$$

$frac(x)$ returns the fractional portion of $x$. Increasing the frequency parameter $\theta$ causes the density of the texture to increase accordingly. Fig. 6.13(b-j) show the reconstructions obtained by applying voxel coloring for increasing values of $\theta$. For comparison, the corresponding texture patterns and the original arc shapes are also shown. In (b), the frequency is so low that the quantized texture pattern is uniform. Consequently, the problem reduces to reconstruction from silhouettes and the result is similar to what would be obtained by volume intersection [MA91, Sze93, Lau95]. Specifically, volume intersection would yield a closed diamond-shaped region; the reconstructed V-shaped cusp surface in (b) corresponds to the set of surfaces of this diamond that are visible from the basis views.

Doubling $\theta$ results in a slightly better reconstruction consisting of two cusps, as shown in (c). Observe that the reconstruction is accurate at the midpoint of the arc, where a texture discontinuity occurs. Progressively doubling $\theta$ produces a series of more accurate reconstructions (d-h) with smaller and smaller cusps that approach the true shape. When $\theta$ exceeds a certain point, however, the reconstruction degrades. This phenomenon, visible in (i) and (j), results when the projected texture pattern exceeds the resolution of the basis images, i.e., when the Nyquist rate is exceeded. After this point, accuracy degrades and the reconstruction ultimately breaks up.

Fig. 6.13 illustrates the following two points: (1) reconstruction accuracy is strongly dependent upon surface texture, and (2) the errors are highly *structured*. To elaborate on the second point, reconstructed voxels drift from the true surface in a predictable manner as a function of local texture density. When the texture is locally homogeneous, voxels drift *toward* the camera volume. As texture density increases, voxels move monotonically away from the camera volume, toward the true surface. As texture density increases even further, beyond the limits of image resolution, voxels continue to move away from the cameras, and away from the true surface as well, until they ultimately disappear.

We next tested the performance of the algorithm with respect to additive image noise. To simulate noise in the images, we perturbed the intensity of each image pixel independently by adding a random value in the range of $[-\sigma, \sigma]$. To compensate, the error threshold was set to $\sigma$. Fig. 6.14 shows the resulting reconstructions. The primary effect of the error and corresponding increase in the threshold was a gradual drift of voxels away from the true surface and toward the cameras. When the error became exceedingly large, the reconstruction ultimately degenerated to the “no texture” solution shown in Fig. 6.13(b). This experiment indicates that image noise, when compensated for by increasing the error threshold, also leads to structured reconstruction errors; higher levels of noise cause voxels to drift progressively closer to the cameras.

The final experiment evaluated the effects of increasing the voxel size on reconstruction accuracy. In principle, the voxel coloring algorithm is only correct in the limit, as voxels become infinitesimally small. In particular, the layering strategy is based on the assumption that points within a layer do not occlude each other. For very small voxels this no-occlusion model is accurate, up to a reasonable approximation. However, as voxels increase in size, the model becomes progressively less accurate. It is surprising, therefore, that the algorithm appears to produce good results even for very large voxel sizes, as seen in Fig. 6.9(d-e).

To more carefully observe the effects of voxel size, we ran the voxel coloring algorithm on the scene in Fig. 6.13(a) for a sequence of increasing voxel sizes. Fig. 6.14 shows the results—the reconstructions are close to optimal, up to the limits of voxel resolution, independent of voxel size. Again, this empirical result is surprising, given the obvious violation of the layering property which is the basis of the algorithm. Some effects of this violation are apparent; some voxels are included in the reconstruction that are clearly invisible, i.e., totally occluded by other voxels from the basis views. For instance, observe that in the reconstruction for voxel size = 10, the top-left and top-right voxels could be deleted without affecting scene appearance from the basis views. These extra voxels are artifacts of the large voxel size and the violation of the layering property. However, these effects are minor and

| Grid Dimensions | Voxels Evaluated | Voxels Colored | Run Time | Reprojection Error |
|---|---|---|---|---|
| $20 \times 24 \times 29$ | 13,920 | 902 | 3 sec | 9.38% |
| $41 \times 49 \times 58$ | 116,522 | 4,898 | 11 sec | 8.01% |
| $83 \times 99 \times 116$ | 953,172 | 21,174 | 62 sec | 7.48% |
| $166 \times 199 \times 233$ | 7,696,922 | 71,841 | 435 sec | 7.20% |

Table 6.2: Voxel Resolution Effects. This table compares the size, run time, and reprojection error for voxel colorings of the dinosaur toy using different grid sizes. Each row represents an 8-fold increase in the number of voxels relative to the previous row. The corresponding models are shown in Fig. 6.9.

do not adversely affect view synthesis in that adding these voxels does not change scene appearance for viewpoints close to the input images.

## 6.6 Discussion

This chapter addressed the problem of view synthesis from numerous basis views distributed widely about a scene. This problem is especially challenging due to the difficulty of computing reliable correspondence information from views that are far apart. A main goal was to determine intrinsic ambiguities and limitations of what is reconstructible, and also to derive a practical algorithm for correspondence computation and view synthesis.

A primary contribution of this chapter was the *voxel coloring* framework for analyzing ambiguities in image correspondence and scene reconstruction. A similar theory was previously developed for the special case of volume intersection, i.e., reconstruction from silhouettes [Lau95, KD95b, Lau97]. The results in this chapter can be viewed as a generalization of the volume intersection problem for the broader case of textured objects and scenes. The voxel coloring framework enabled the identification and computation of *color invariants*—points having the same color in every possible scene reconstruction, consistent with a set of basis images.

A second important contribution was the voxel coloring algorithm for computing pixel correspondence from a set of basis images. A key element was the *ordinal visibility constraint*, a novel constraint on the configuration of camera viewpoints that enabled an efficient solution to the correspondence problem. This is the first practical algorithm capable of generating provably-consistent dense correspondence maps from a set of input images, in the presence of occlusion. It is therefore useful not just for view synthesis, but for other applications that require correspondence, e.g., motion analysis and 3D scene reconstruction.

The algorithm has several novel features that make it especially attractive for view synthesis tasks:

- **Generality**: The low-level voxel representation can approximate any surface type and easily models discontinuities. It is therefore well-suited for modeling complex real-world objects and scenes.

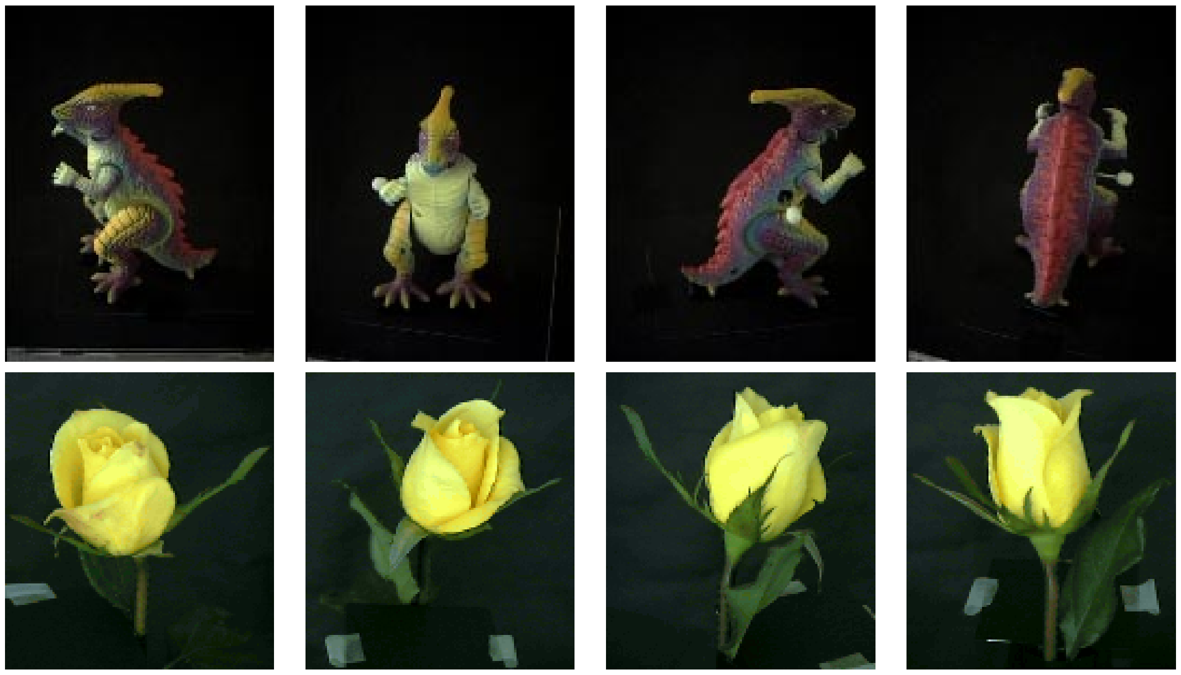


Figure 6.7: Selected basis images for a dinosaur toy (top) and a rose (bottom). 21 images were taken in all, spanning close to a $360°$ rotation of the object.

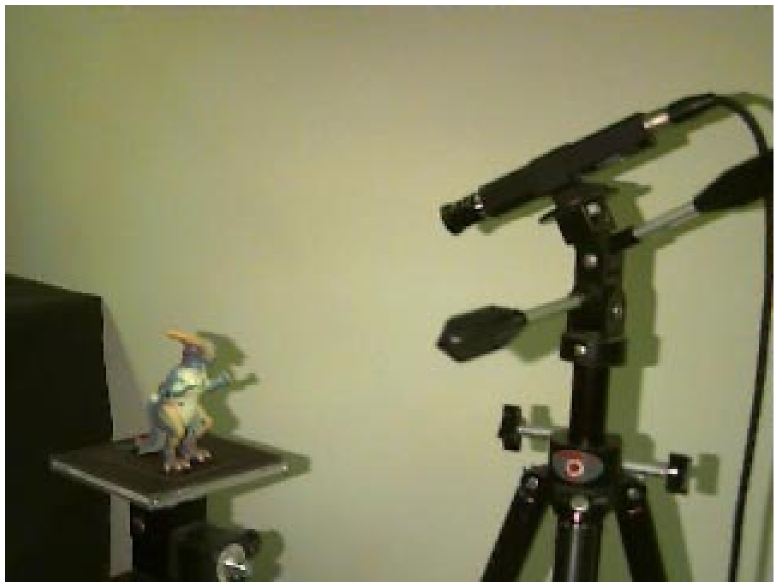
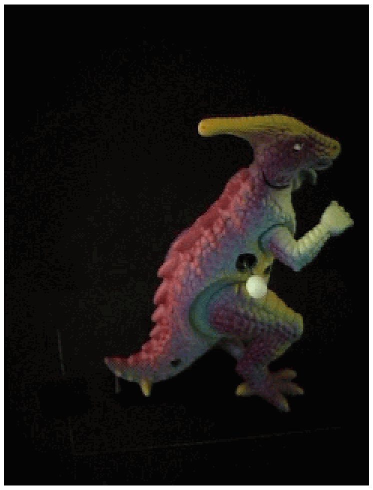
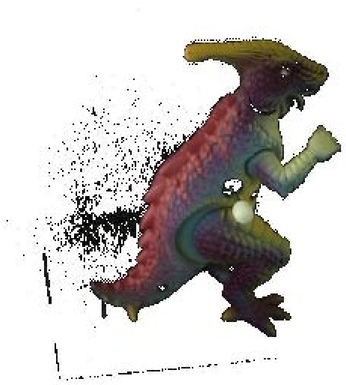


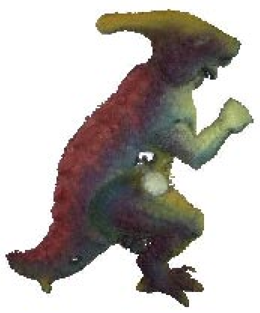
Figure 6.8: Image Acquisition Setup. Basis images were captured by placing an object on a calibrated pan-tilt head and rotating the object in front of a stationary video camera. The camera was placed above the object facing downwards to satisfy the ordinal visibility constraint.



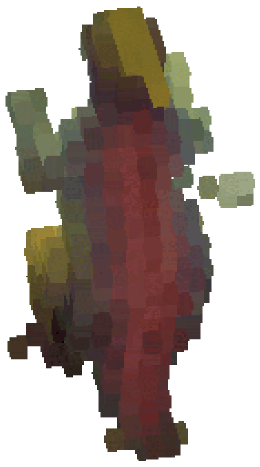
**Input Image**
**(a)**



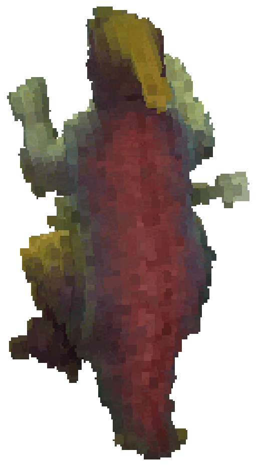
**Thresholded Image**
**(b)**



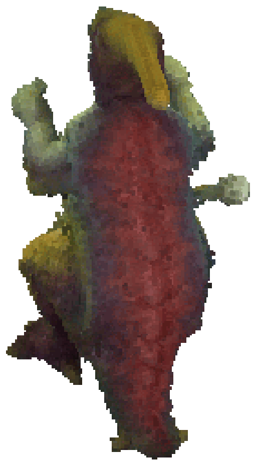
**Model Reprojection**
**(c)**



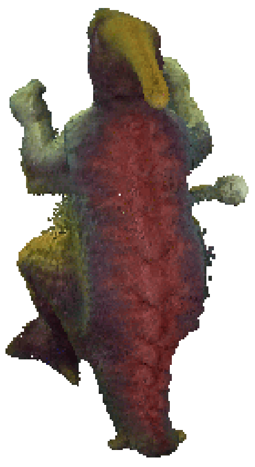
**902 voxels**
**(d)**



**4,898 voxels**
**(e)**

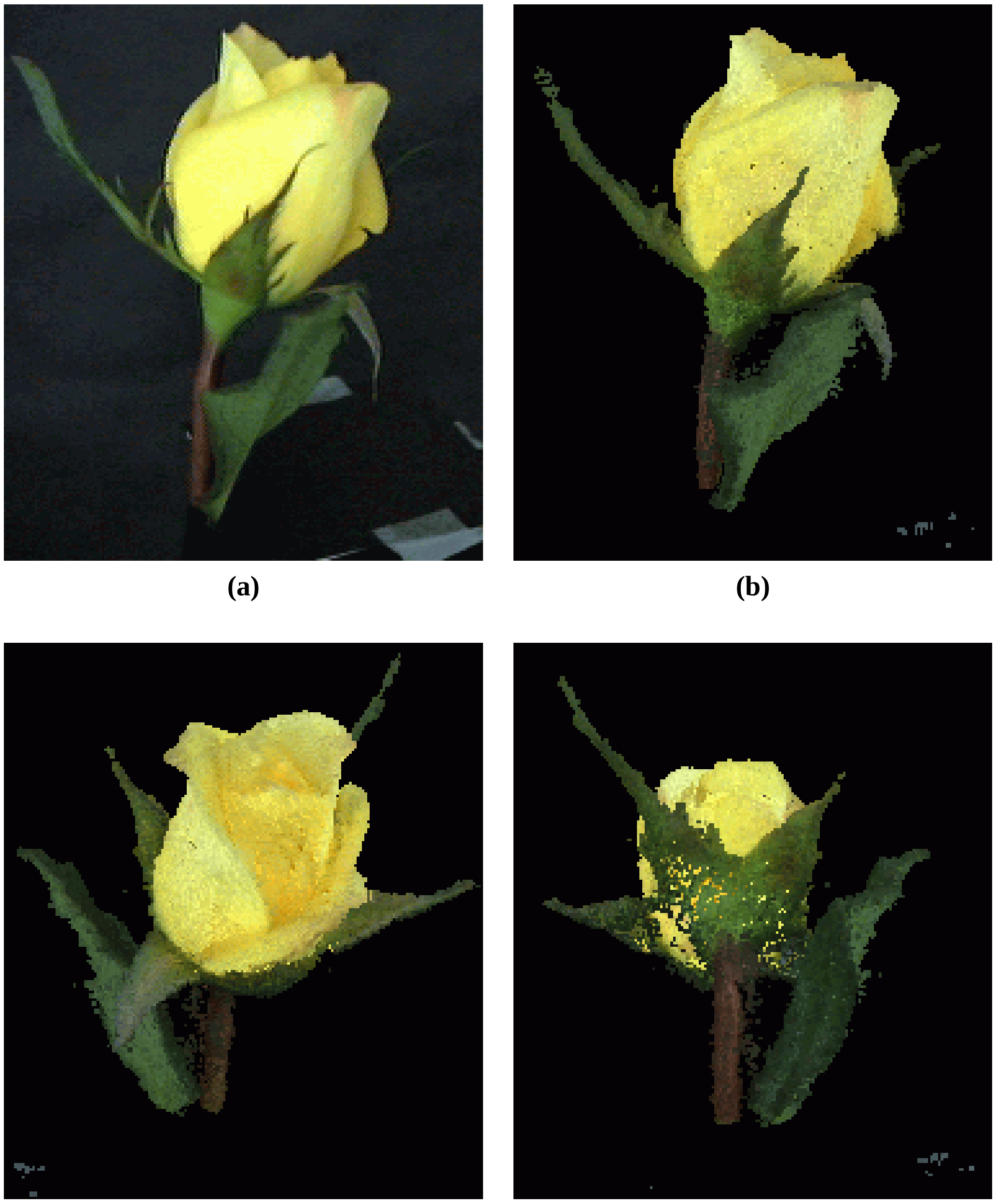


**21,174 voxels**
**(f)**



**71,841 voxels**
**(g)**

Figure 6.9: Voxel Coloring of a Dinosaur Toy. Original image (a) is thresholded (b) to eliminate most of the background pixels. The voxel coloring (c) was computed from 21 thresholded images of the object undergoing a 360° rotation. (d-g) show the reconstruction from a new viewpoint at different voxel resolutions, where the voxel width is progressively doubled.



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

Figure 6.10: Voxel Coloring of Flower. Input image (a) is shown next to the projection of a voxel coloring for same viewpoint (b). The reconstruction (70K voxels) captures image appearance very accurately for new views, such as (c), that are near the input viewpoints. Artifacts become more prevalent when the new viewpoint is far away from the input views, as in (d).

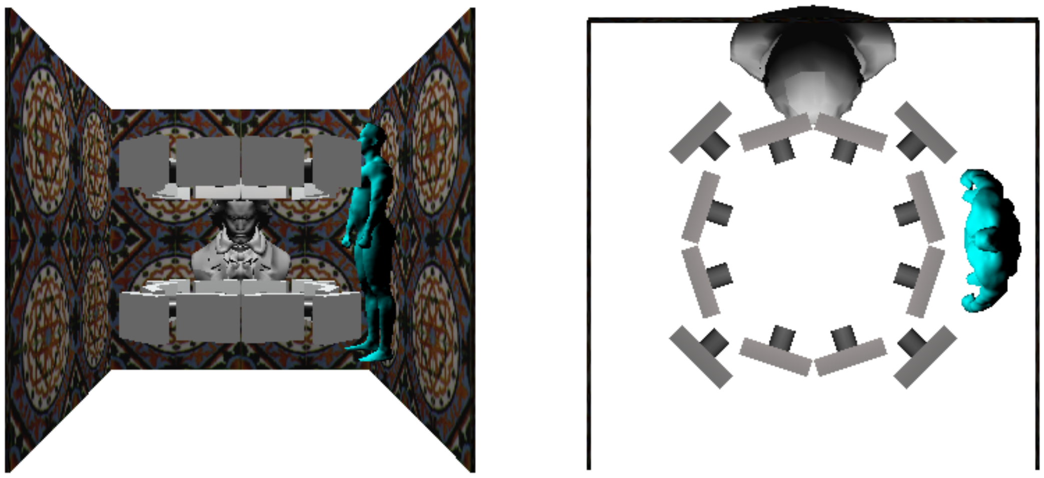


Figure 6.11: Placement of input camera viewpoints for a *panoramic* synthetic room scene. The camera positions and scene are shown together, from a frontal (left), and overhead (right) perspective.

- **Flexible Acquisition**: The cameras may be arbitrarily far apart without degrading reconstruction accuracy. Indeed, the algorithm performs best when cameras are distributed widely about the scene.

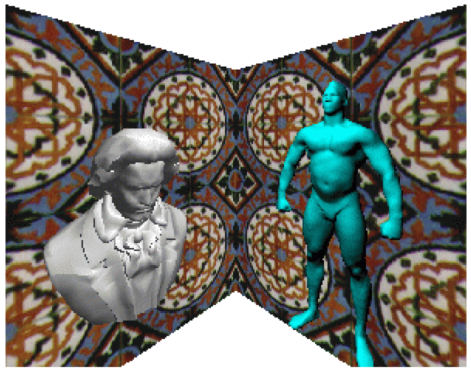
- **Panoramic Visibility**: The voxel coloring method can synthesize views for any camera position and orientation. The fact that it is applicable for both inward- and outward-facing camera configurations makes it ideally suited for panoramic scenes such as room interiors.

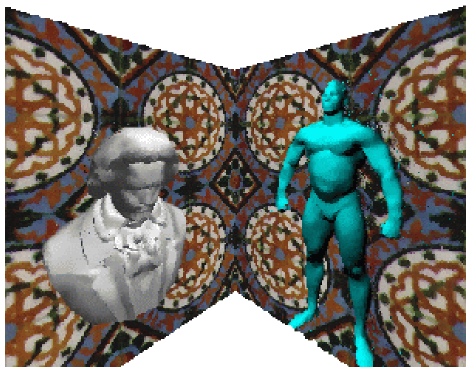
- **Insensitivity to Occlusion**: Changes in visibility are fully modeled by the algorithm and impose no performance penalty.

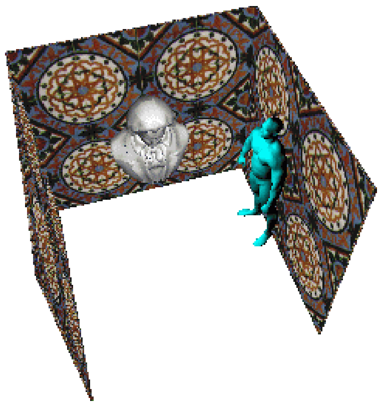
- **Efficiency**: The algorithm performs only a single pass through the scene volume and exploits regular operations that can be performed using texture-mapping graphics hardware. We are currently investigating a real-time implementation of the algorithm that would run on existing graphics workstations. The technique is also space efficient in that only *surface* voxels are stored, i.e., voxels that are visible in at least one basis image.

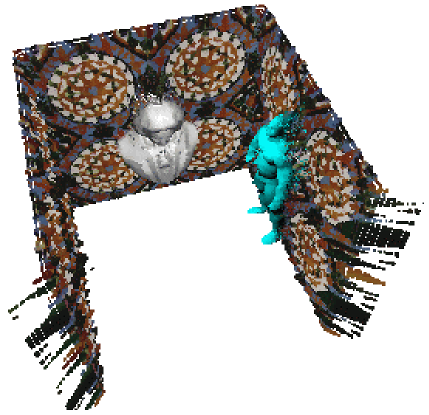
- **Scalability**: By varying the voxel size, the algorithm can be tailored to available computing resources. Empirical evidence demonstrates that the algorithm performs well for a range of different voxel sizes.

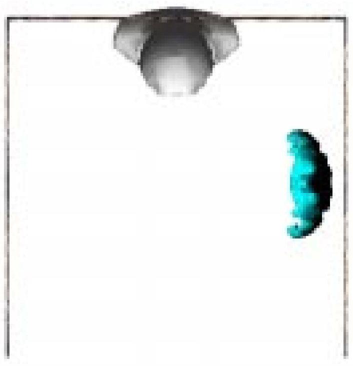
While the *ordinal visibility constraint* is needed to support a single-pass algorithm, it also rules


**(a)**


**(b)**


**(c)**


**(d)**


**(e)**

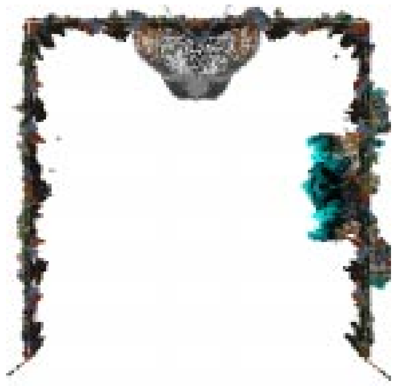

**(f)**

Figure 6.12: Panoramic room scene reconstruction. Renderings of the true scene are shown at left and the reconstructed scene at right. (a) and (b): an input viewpoint, from *inside* the room. (c) and (d): a new viewpoint from above the room. (e) and (f): an extreme overhead view. Fidelity is best near the input viewpoints and degrades smoothly as the camera moves further away.

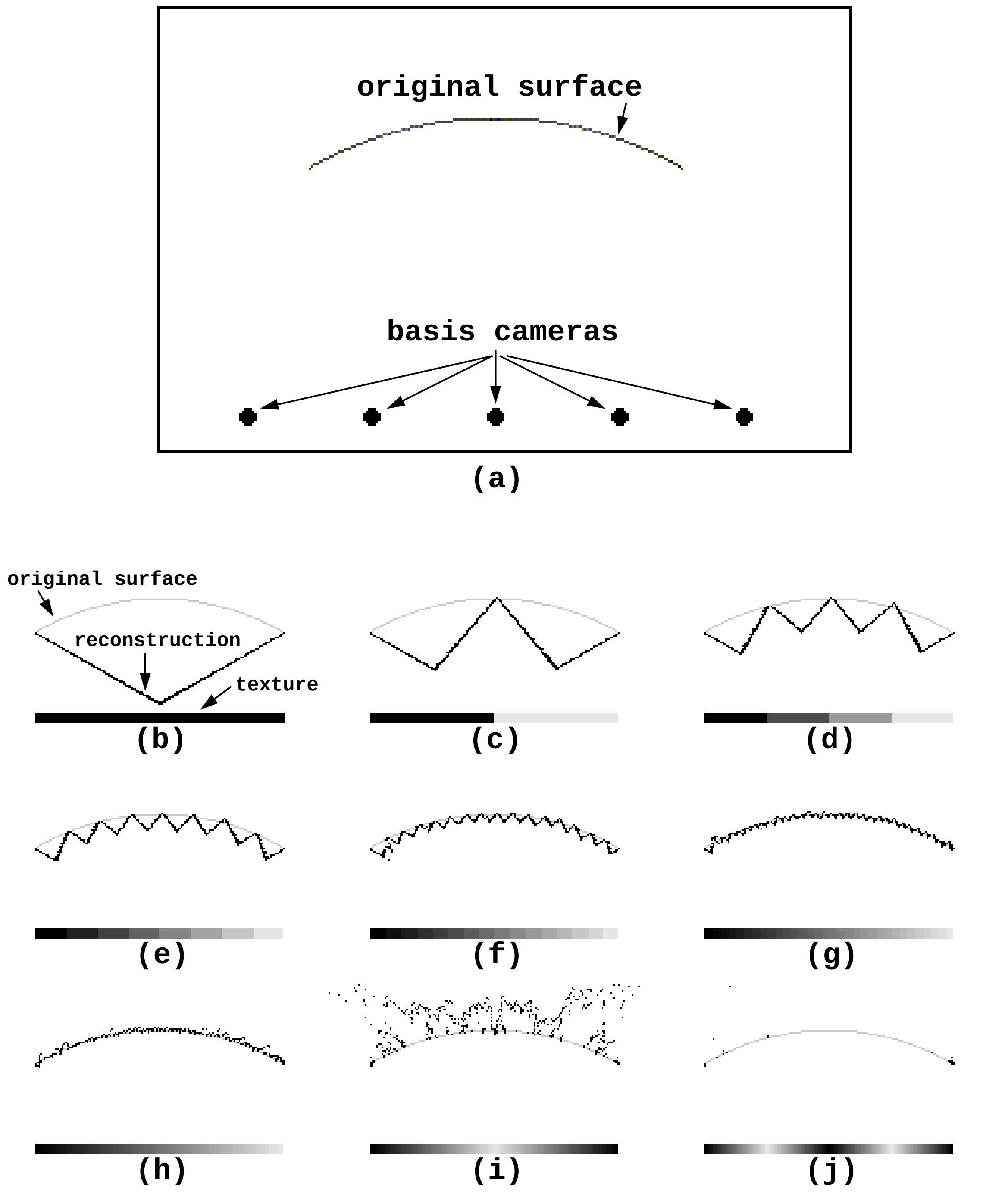


Figure 6.13: Effects of Texture Density on Voxel Reconstruction. (a): A synthetic arc is reconstructed from five basis views. The arc is textured with a cyclic gradient pattern with a given frequency. Increasing the frequency makes the texture denser and causes the accuracy of the reconstruction to improve, up to a limit. In the case of (b), the texture is uniform so the problem reduces to reconstruction from silhouettes. As the frequency progressively doubles (c-j), the reconstruction converges to the true shape, until a certain point beyond which it exceeds the image resolution (i-j).

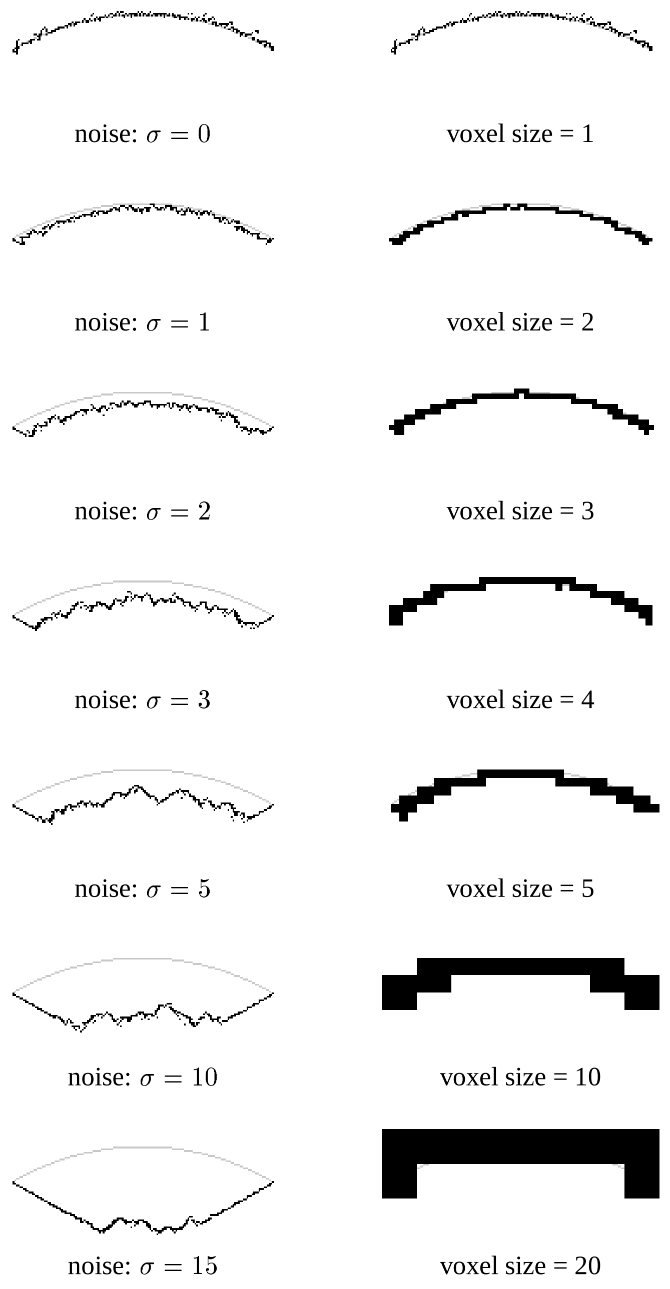


Figure 6.14: Effects of Image Noise and Voxel Size on Reconstruction. Image noise was simulated by perturbing each pixel by a random value in the range $[-\sigma, \sigma]$. Reconstructions for increasing values of $\sigma$ are shown at left. To ensure a full reconstruction, the error threshold was also set to $\sigma$. Increasing noise caused the voxels to drift from the true surface (shown as light gray). The effects of changing voxel size are shown at right. Notice that the arc shape is reasonably well approximated even for very large voxels.

out certain input camera configurations. Although the scene can surround the cameras, as in the room scene in Fig. 6.12, the cameras cannot surround the object or scene. This is not a serious limitation in controlled lab environments where the camera configuration may be designed with the ordinal visibility constraint in mind. For instance, in our experiments the cameras were raised slightly above the scene to be reconstructed (Fig. 6.8). However, it could be problematic for other situations, e.g., it would not allow reconstruction from a video sequence obtained by walking all the way around a large object with a video camera. We are currently investigating methods for handling these types of camera motions and configurations. One solution would be to segment the basis images into sets that individually satisfy the ordinal visibility constraint, run the algorithm on each set separately, and then merge the results. Automatic methods for performing this segmentation task would simplify this approach. Another possible approach would be to extend the voxel coloring algorithm to directly handle general camera configurations, perhaps by using a multi-pass approach. This is a topic for future work.

The results in Section 6.5 indicate that low-contrast regions and noise produce reconstruction errors, e.g., *cusps*, that are highly structured. For view synthesis tasks, this is a potential disadvantage, since structured noise can produce perceptible artifacts. The fact that these errors are deterministic and well-understood indicates that they are potentially detectable, and thus could be attenuated. One solution would be to perform a post-processing phase, analogous to dithering [FS75], in which errors are diffused to mitigate these artifacts. An alternative method would be to identify textureless regions and other error sources in the basis images and treat these features specially in the reconstruction process.

The notion of *color invariance* bears resemblence to the problem of *color constancy* (c.f. [HSW92]) which has a long history in the computer vision literature. The color constancy problem is to determine, from one or more images, a description of scene material properties (e.g., surface reflectance) that does not depend on scene illumination. However, invariance to illumination is quite different than the notion of color invariance used in this chapter; in the latter case, illumination is held *fixed*. Rather than separating reflectance from illumination, color invariants encode scene radiance directly, which is sufficient to synthesize new views of the scene with illumination held constant.

# Chapter 7

# Editing[†]

So far we have described how view synthesis can be implemented as a 2D operation, applied to a set of basis images. It is instructive to consider other types of "3D" operations that can be applied in the image domain. Indeed, traditional 3D graphics modeling systems typically support a number of different capabilities, including

- **View synthesis**: creating images from varying camera viewpoints
- **Illumination synthesis**: changes in scene illumination
- **Editing**: interactive modifications to surface position, shape, and surface properties
- **Animation**: a scripted sequence of the above operations

Most work on image-based representations has focused on the first problem—view synthesis. More recently, there has been increasing attention in modeling reflectance and illumination via analysis of basis images [War92, BK96, DNvGK97, SWI97, SK98]. However, the problems of image-based editing and animation have yet to be studied. This chapter addresses the editing problem and describes several 3D editing operations that can be performed on photographs.

Image editing programs like Adobe Photoshop provide ways of modifying an object's appearance in a single image by manipulating the pixels of that image. Ultimately, however, one might like to visualize how edits to an object in one image affect its appearance from other viewpoints. For instance, consider choosing wallpaper for a room in your house by painting the wallpaper pattern into one of several digitized photographs of the room. As you paint a wall in one image, the pattern appears instantly at the appropriate place in the other photographs, providing feedback on how the modified room would look from several different viewpoints. Similarly, scissoring out an object (e.g., a vase) from one or two frames of a video walkthrough of a room could remove that object from the entire video by automatically propagating the scissoring operation to the other images. Additional controls could modify camera viewpoint, allowing the effects of image edits to be visualized from viewpoints other than those of the room's original photographs.

A key feature of these types of editing operations is that they apply to the space of all views of the scene, rather than just one image. It is therefore convenient to cast them in terms of the *plenoptic*

[†] The material presented in this chapter is joint work with Kiriakos N. Kutulakos at the University of Rochester. The main ideas came about as the result of discussions between him and the author.

*function* [AB91, MB95b], which encodes scene appearance from all possible viewpoints. Within this framework, our goal is to recover a scene's plenoptic function from a discrete set of images and to determine how it should be modified in response to basic image editing operations like painting, scissoring, and morphing. We use the term *plenoptic* to describe image editing operations that modify the plenoptic function and can therefore be propagated to new viewpoints.

## 7.1 The User View: Editing by Example

Plenoptic image editing is an approach that allows a user to virtually modify an object's appearance by editing any of several photographs of the object at different positions and orientations. From the point of view of the user, all interaction occurs via manipulations to *individual images* using conventional pixel-editing tools. The user simply specifies how one or more images should look by painting and moving pixels until the desired look is achieved. Pixel modifications by the user are interpreted as new constraints on object appearance that induce changes to the plenoptic function and therefore affect every image. In this way, user edits of a single image can be propagated to other images of an object.

To the user, a plenoptic image editing system appears very similar to current image editing programs like Photoshop. Pixels of one or more images are edited by direct manipulation using a standard suite of painting, scissoring (cut and paste), and warping tools found in many image editing programs. In fact, if only one image is on screen, there is no visible difference between a conventional image editing program and the plenoptic version. The difference becomes apparent, however, when two or more images are viewed side by side. Any change to a region or object in one image is instantly propagated to the corresponding part in the other image(s). For instance, removing a freckle in one of several photographs of a face causes the freckle to disappear simultaneously from all other images. In this way, the propagation mechanism can be used as a kind of *power-assist*—the user can affect many different images of an object by editing only one or two.

The freckle example illustrates the basic model for plenoptic image editing: a user specifies how regions in one or more images should look *by example*, and the system determines how to consistently propagate the modifications to the other images. This editing-by-example model provides a very powerful way for the user to control object appearance *plenoptically*, i.e., in all views at once, by editing a small number of images in a direct, intuitive way.

In the remainder of this section, we discuss plenoptic versions of some standard image-editing operations. The list is not meant to be comprehensive, but provides examples of what different types of image editing operations can do within a plenoptic framework. The implementation of these operations is discussed in Section 7.2.

### 7.1.1 Plenoptic Painting

A basic type of image editing operation is to change pixel colors by drawing over an image region with a digital paintbrush. In the plenoptic framework, a paint operation is interpreted as a modification to the material properties of the surface points whose projection coincides with the painted region. The change therefore affects every image of the object, and properly accounts for differences in visibility between views. The multi-image updates appear in real time, allowing the user to fluidly paint in several images simultaneously by moving a brush over one image. Fig. 7.1(b) and (f) show images from a real-time plenoptic paint operation in action.

### 7.1.2 Plenoptic Scissoring

An image scissoring operation eliminates or extracts a set of regions from an image, often for inclusion in a different image. Whereas an image-scissoring operation extracts a region of an image, a plenoptic image scissoring operation carves out part of the plenoptic function, causing a corresponding region to be extracted in every image. Scissoring out the image region therefore has the effect of cutting out the portion of the visible object that projects to that image region.

Plenoptic scissoring enables some interesting effects that are not possible with regular scissoring. For instance, it is possible to "see through" objects in an image by scissoring them out and exposing what lies behind. This capability is shown in Fig. 7.1(g) and is achieved by extrapolating the appearance of hidden surfaces from other images in which those surfaces are visible, using the derived plenoptic model. The extrapolation occurs automatically whenever the user performs a scissoring operation.

### 7.1.3 Plenoptic Morphing

Image warping or *morphing* is a popular way of producing shape changes and animations from one or more images. Although multi-image morphs can be performed in the plenoptic framework, we restrict our attention to the case in which a single image is warped by displacing individual pixels using a 2D motion flow field. Instead of warping pixels, a plenoptic morph warps the underlying 3D scene so as to be projectively consistent with the warped image. The shape change is thereby propagated to other images automatically. Fig. 7.1(d) and (h) show an application of a plenoptic image warp.

The effects of the image warp on the underlying scene structure can be thought of in terms of warping *rays* rather than pixels. Consider the ray that originates at a camera's center and passes through the image plane at a particular pixel. Moving this pixel corresponds to moving all points along the ray to coincide with the ray passing through the destination pixel.

### 7.1.4 New View Generation

In addition to propagating changes between images, the plenoptic framework can generate arbitrary new views of an object by evaluating the recovered plenoptic function at new user-specified viewpoints. The synthetic views automatically incorporate the modifications incurred by user edits to the images, since these edits induce changes to the plenoptic function.

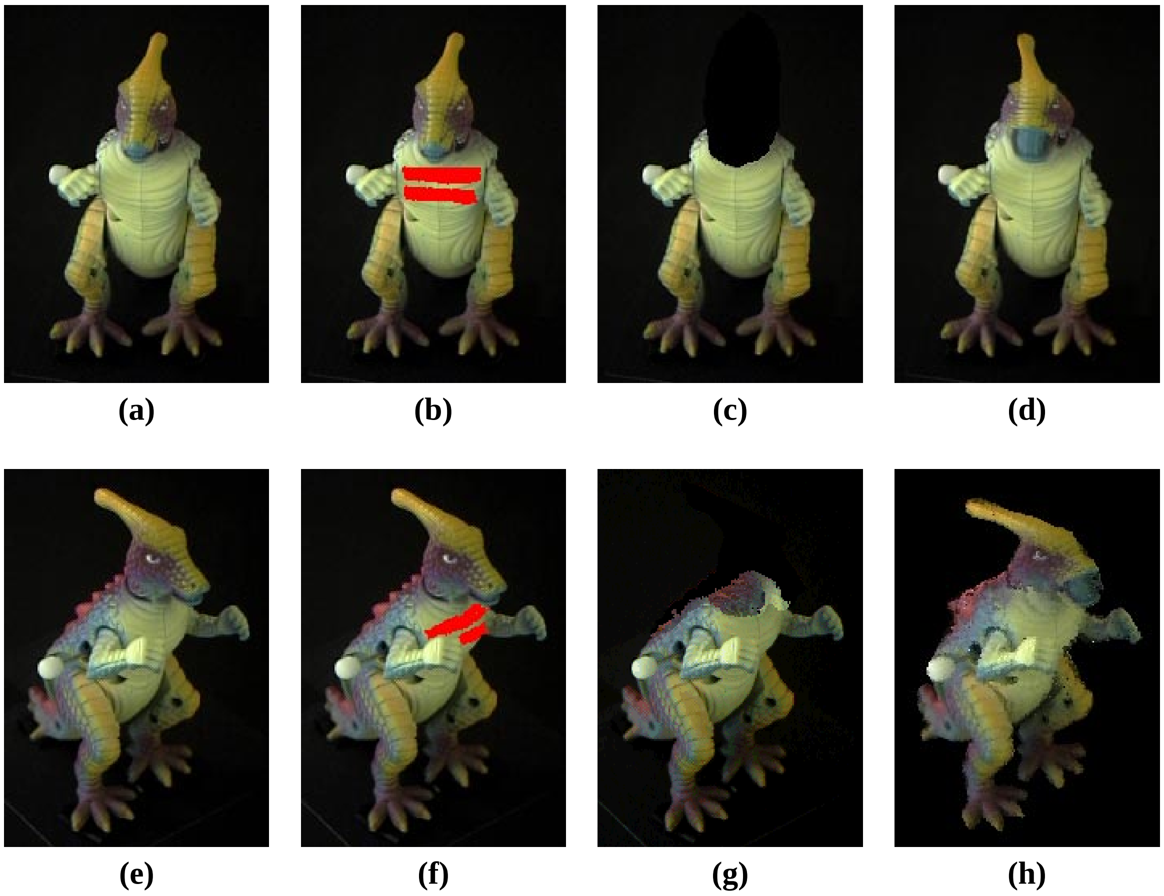


Figure 7.1: Examples of plenoptic image editing operations applied to photographs of a dinosaur toy. (b)-(d) show image painting, scissoring, and morphing operations, respectively, applied to image (a). (f)-(h) show images that were automatically generated by propagating the respective editing operations to image (e). Observe that the propagation properly accounts for differences in visibility between the two views—part of the painted area is correctly occluded by the dinosaur’s right hand in (f), and cutting off the head in (c) exposes surfaces in image (g) that were not visible in the original image (e). These new surfaces are *synthesized* from other viewpoints so that (g) represents a composite of a real photograph with synthesized image regions.

The ability to generate new views is also useful for edit operations, because it allows the user to interactively choose a good image for editing. For instance, a flat surface can be rotated to a front-on view to facilitate painting and avoid foreshortening affects. Similarly, scissoring a region is easier when the entire region is visible in a single image.

## 7.2 Edit Propagation

In order to support plenoptic editing operations, we must first model the plenoptic function in a way that enables modifications via changes to a single image. Existing image-based representations were designed primarily for view synthesis, not for editing, and some cannot be easily extended to support editing operations. For instance, a recent direction in image-based view synthesis has been toward ray-based plenoptic representations, in particular the light field [LH96] and Lumigraph [GGSC96, SCG97], in which light rays are represented separately. An advantage of these models is that views can be synthesized without having to compute a dense pixel correspondence. Performing image editing operations within a ray-based representation is difficult, however, for two reasons. First, local image modifications can affect object appearance from disparate views and may therefore require global changes to a ray-based representation. Second, the lack of correspondence information makes it difficult to propagate editing operations between images.

Edit propagation therefore calls for a richer representation of the plenoptic function that encodes correspondence information. In [SK98], we describe such a representation, called *plenoptic decomposition*, that decomposes the plenoptic function into shape and radiance components, thereby enabling view synthesis, illumination synthesis, and a range of plenoptic image editing operations. A full discussion of this representation is beyond the scope of this thesis. However, a subset of this functionality, including some editing operations, is obtainable with only correspondence information, such as that provided by image morphing tools or the automatic methods described in Chapter 6.

In the remainder of this section, we describe how the plenoptic image editing operations shown in Fig. 7.1 were implemented, using voxel coloring to compute the requisite correspondence information.

### 7.2.1 Painting

Painting is the simplest of the plenoptic image editing functions because it changes only the color of scene voxels without any modification to shape. Propagating a painted pixel requires first determining the voxel(s) that corresponds to that pixel and modifying its color. The change is then propagated by projecting the voxel(s) into each image in which it is visible and recoloring corresponding pixels in those images.

Plenoptic painting can be performed in real-time by precomputing the mapping between pixels in each image and voxels in the voxel coloring. This can be achieved using two lookup tables: $\overline{\mathcal{S}}(\mathbf{p})$ returns the voxel(s) visible at pixel position $\mathbf{p}$ and $\mathcal{I}(\mathbf{V})$ returns the pixel(s) $\mathbf{p} \in \mathcal{I}$ for which $\mathbf{V} = \overline{\mathcal{S}}(\mathbf{p})$. This enables propagating pixel edits simultaneously to $n$ images at the cost of $n$ table lookups per painted pixel in the original image. Using this method, our implementation provides paint propagation at interactive rates.

### 7.2.2 Scissoring

Image scissoring extracts a set of pixels from an image. Similarly, plenoptic scissoring removes a set of voxels from the voxel coloring. One option is to remove the set of voxels that project unoccluded to the affected pixels. This may expose new voxels in the scissored image, behind those that were removed. Alternatively, scissoring can remove all voxels that project to the affected pixels, whether or not they are visible. The latter method was used to generate the images in Fig. 7.1(c) and (g).

Performing the propagation requires masking out pixels in each image that correspond to the projection of voxels removed by the scissoring operation. These pixels are then filled in by rendering the modified voxel coloring from the same viewpoint as the edited photograph and copying these pixels from the rendered image. The resulting image therefore includes a mixture of pixels from both the original photograph and the synthesized view.

### 7.2.3 Morphing

As described in Section 7.1.3, an image morph induces a warping of scene rays. Consider the set of rays passing from a camera center through the image plane. An image morph deforms the image plane, causing these rays to move with it. In turn, the motion of a ray moves all scene voxels that lie on the ray. While the motion of rays is determined, the motion of voxels along rays is not. Our implementation of plenoptic image morphing fixed this variable by constraining voxels to move parallel to the image plane.

We use an implementation of the Beier and Neely method [BN92] to generate image warps. The voxel warp is computed by inverse-mapping each voxel in the warped scene to determine the corresponding voxel (if any) in the unwarped scene. Once the voxel warp is computed, it is propagated to other images by re-rendering the images using the warped set of voxels.

# Chapter 8

# Conclusions

This thesis investigated the problem of rendering changes in viewpoint and scene appearance by operating on a set of basis images of a real scene. A primary focus was the *view synthesis problem*, i.e., producing images of a real scene from new viewpoints. An additional topic was *editing transformations* in which persistent changes to scene appearance could be effected by editing individual basis images.

## 8.1 Contributions

The view synthesis problem requires extracting information from a set of input images of a scene in order to synthesize one or more output images, where the output images correspond to physically-consistent views of the scene. Achieving this goal required answering fundamental questions relating to measurability and uniqueness. In particular, under what conditions are the images sufficient to predict new views, and which views are determined? In response to these questions, this thesis made the following contributions:

- **A uniqueness result for the two-view case**. This result proved the following: given two views of a static Lambertian scene satisfying *monotonicity* (i.e., no occlusions), all *in-between* views on the line segment between the two camera centers are uniquely determined. Importantly, this result required neither pixel correspondence nor known camera positions. It therefore demonstrates that the view synthesis problem is *well-posed* under monotonicity. In contrast, the shape reconstruction problem is known to be *ill-posed* under the same conditions [PTK85].

- **Boundary flow**. View synthesis requires computing correspondence information from the two basis views. However, the true optical flow field is not measurable from the two basis views [PTK85]. An important contribution of this thesis was introducing a new type of flow field called *boundary flow* that is both measurable and sufficient for view synthesis.

- **The voxel coloring framework**. Using this framework, a new type of scene invariant, called *color invariant*, was derived, corresponding to a point that has the same radiance characteristics in every consistent scene reconstruction. Color invariants are useful first because they provide intrinsic scene information and second because the collection of all such points was shown to form a complete scene reconstruction.

- **The ordinal visibility constraint**. A novel constraint on the relative positions of $n$ basis cameras determined a fixed visibility order for points in the scene. The key feature of this visibility order is that it is constant for all cameras and does not depend on the structure of the scene. Consequently, occlusion relationships are greatly simplified.

These contributions provide a rich theoretical basis for studying view synthesis algorithms and applications. A primary focus was the development of practical algorithms for view synthesis and other image-based scene transformations. These algorithms were developed within this theoretical framework, but were designed to be robust with respect to deviations from the assumptions. The main practical contributions were:

- **The view morphing algorithm**. This algorithm used concepts from image morphing and projective geometry to synthesize physically-consistent in-between views from a pair of uncalibrated basis images. The technique can operate either fully automatically, using adaptations of stereo techniques to compute boundary flow, or can exploit user interaction to provide sparse correspondence information. The latter method of correspondence yielded especially good results and was proven to be highly robust with respect to occlusions, changes in scene structure, and variations in illumination. Extensions were outlined that enabled view synthesis from a single view and from three or more views.

- **The voxel coloring algorithm**. This algorithm exploits the ordinal visibility constraint to compute a voxel reconstruction and dense pixel correspondence maps from a set of basis images. The algorithm is noteworthy in that it generates reconstructions that are fully consistent with the basis images, for cameras spaced arbitrarily far apart. Furthermore, the method has the following advantages: (1) it generates dense reconstructions and correspondence maps from any number of basis images, (2) performance does not degrade under occlusion or changes in the field of view, (3) it enables *panoramic* reconstructions and visualizations, (4) it is efficient and amenable to hardware acceleration on existing graphics workstations, (5) it performs well for a range of different voxel sizes and therefore can be easily scaled to match available computation resources, and (6) it produces high-quality reconstructions and synthesized views from images of complex natural scenes.

- **The plenoptic image editing framework**. A new class of interactive image editing operations was introduced, designed to maintain consistency between multiple views of the scene. The distinguishing feature is that edits to any one image propagate automatically to all other images as if the scene had itself been modified. The method used the voxel coloring algorithm to derive correspondence information, attesting to the usefulness of voxel coloring for facilitating a range of different image-based scene transformations.

## 8.2 Limitations and Future Work

Several assumptions were used in this thesis to simplify the analysis. Chief among these is the assumption that scene surfaces are Lambertian, which was used throughout as a way to simplify the correspondence problem. This model, however, does not take into account reflections and specularities that are prevalent in most real scenes. An important topic of future work will be to develop view synthesis algorithms and other image-based transformations that model non-Lambertian effects.

One potential application of this work is to construct image-based models of real objects that can be easily acquired from photographs and combined with other graphical objects in a virtual scene. For instance, this would enable *virtual studios* for film production in which images of real actors could be composited into realistic image-based scenes. Observe, however, that the view synthesis approaches in this thesis assumed fixed illumination and are capable of synthesizing only views with the same illumination. This constraint does not permit combining objects acquired from different sets of basis views together into a common scene, unless they are acquired under identical illumination. A useful capability would therefore be *illumination synthesis*, i.e., processing a set of basis images to synthesize images of the scene under new lighting configurations. One approach would be to model surface reflectance in order to predict the radiance from each voxel as a function of viewpoint and illumination. Illumination synthesis would then require varying the illumination in the basis images and reconstructing the reflectance function for each voxel. This idea is beyond the scope of this thesis, but is investigated further in [SK98].

Another application of interest is to apply view synthesis methods for large-scale environments, to provide walk-throughs of buildings such as museums, or visual exploration of entire cities. The voxel coloring approach is particularly suited for this type of application, based on its generality and ability to provide panoramic visualizations. However, a key limitation is the need for calibrated images. Current technology does not easily enable acquiring precisely-calibrated images of large environments. Creating visualizations of these environments therefore calls for further research into calibration techniques for large image sequences as well as new view synthesis methods that work with uncalibrated images.

A complete solution to the view synthesis problem requires answering questions such as: *which* views can be uniquely predicted from a set of basis images, and *how* may new views be synthesized? In developing view synthesis algorithms, virtually all work in this field has focused only on the latter question. This thesis also made important steps toward answering the former question by establishing the uniqueness result and introducing the voxel coloring framework. Significant work remains to be done, however, before this question is completely answered. Under general conditions, it is unlikely that all views can be unambiguously predicted from a discrete number of basis views, given the *aperture problem* and other fundamental limitations. However, it may be possible to formulate assumptions like monotonicity, but which are less restrictive, that enable unique prediction of a specific range of views. In addition, one could attempt to characterize the set of all possible reconstructions that are consistent with a set of basis images. This set could be used to assess the variability of shape and color in scene reconstruction and to derive strict error bounds for synthesized images.

An important feature of the view synthesis algorithms described in this work is that they can produce very realistic images even when basic assumptions like constant illumination, rigid shape, and Lambertian surfaces are violated. While these images may not correspond precisely to physically-correct changes in viewpoint, they convey visually convincing camera transformations and may therefore be sufficient for applications like visualization. Given these results, it would be useful to expand the theory to include a perceptual model of correctness of synthesized views. One important use would be to quantify the conditions under which view morphing, voxel coloring, and other view synthesis algorithms yield images that are visually convincing, i.e., that are sufficient to convey realistic camera transformations. In addition, such a model could facilitate the development of a new class of algorithms that exploit properties of human perception to synthesize realistic, non-physical views with greater efficiency.

# Bibliography

[AB91] Edward H. Adelson and J. R. Bergen. *The Plenoptic Function and the Elements of Early Vision*. MIT Press, Cambridge, MA, 1991.

[AL91] Nicholas Ayache and Francis Lustman. Trinocular stereo vision for robotics. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 13(1):73–85, 1991.

[App95] Apple Computer, Inc. QuickTime VR, version 1.0, 1995.

[AS95] Serge Ayer and Harpreet S. Sawhney. Layered representation of motion video using robust maximum-likelihood estimation of mixture models and mdl encoding. In *Proc. Fifth Int. Conf. on Computer Vision*, pages 777–784, 1995.

[AS97] Shai Avidan and Amnon Shashua. Novel view synthesis in tensor space. In *Proc. Computer Vision and Pattern Recognition Conf.*, pages 1034–1040, 1997.

[BA83] P. J. Burt and E. H. Adelson. The laplacian pyramid as a compact image code. *IEEE Trans. Communications*, 31:532–540, 1983.

[BA96] Michael J. Black and P. Anandan. Estimating optical flow in segmented images using variable-order parametric models with local deformations. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 18(10):972–986, 1996.

[Bas92] Ronen Basri. On the uniqueness of correspondence under orthographic and perspective projections. In *Proc. Image Understanding Workshop*, pages 875–884, 1992.

[BB89] H. Harlyn Baker and Robert C. Bolles. Generalizing epipolar-plane image analysis on the spatiotemporal surface. *Int. J. of Computer Vision*, 3(1):33–49, 1989.

[BBM87] Robert C. Bolles, H. Harlyn Baker, and David H. Marimont. Epipolar-plane image analysis: An approach to determining structure from motion. *Int. J. of Computer Vision*, 1(1):7–55, 1987.

[BCZ93] Andrew Blake, Rupert Curwen, and Andrew Zisserman. A framework for spatiotemporal control in the tracking of visual contours. *Int. J. of Computer Vision*, 11(2):127–145, 1993.

[BFB94] J. L. Barron, D. J. Fleet, and S. S. Beauchemin. Performance of optical flow techniques. *Int. J. of Computer Vision*, 12(1):43–77, 1994.

[BK96] Peter N. Belhumeur and David J. Kriegman. What is the set of images of an object under all possible lighting conditions. In *Proc. Computer Vision and Pattern Recognition Conf.*, pages 270–277, 1996.

[Bla97] Black Diamond, Inc. Surround Video, version 1.0, 1997.

[BM92] Peter N. Belhumeur and David Mumford. A Bayesian treatment of the stereo correspondence problem using half-occluded regions. In *Proc. Computer Vision and Pattern Recognition Conf.*, pages 506–512, 1992.

[BN92] Thaddeus Beier and Shawn Neely. Feature-based image metamorphosis. In *Proc. SIGGRAPH 92*, pages 35–42, 1992.

[BP93] Roberto Brunelli and Tomaso Poggio. Face recognition: Features versus templates. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 15(10):1042–1052, 1993.

[BP96] David Beymer and Tomaso Poggio. Image representations for visual learning. *Science*, (272):1905–1909, 1996.

[BTZ96] Paul Beardsley, Phil Torr, and Andrew Zisserman. 3D model acquisition from extended image sequences. In *Proc. European Conf. on Computer Vision*, pages 683–695, 1996.

[CB92] R. Cipolla and A. Blake. Surface shape from the deformation of apparent contours. *Int. J. of Computer Vision*, 9(2):83–112, 1992.

[Che95] Shenchang Eric Chen. Quicktime VR — An image-based approach to virtual environment navigation. In *Proc. SIGGRAPH 95*, pages 29–38, 1995.

[CL96] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proc. SIGGRAPH 96*, pages 303–312, 1996.

[Col96] Robert T. Collins. A space-sweep approach to true multi-image matching. In *Proc. Computer Vision and Pattern Recognition Conf.*, pages 358–363, 1996.

[Col97] Robert T. Collins. Multi-image focus of attention for rapid site model construction. In *Proc. Computer Vision and Pattern Recognition Conf.*, pages 575–581, 1997.

[Cox93] Ingemar J. Cox. A review of statistical data association techniques for motion correspondence. *Int. J. of Computer Vision*, 10(1):53–66, 1993.

[CW93] Shenchang Eric Chen and Lance Williams. View interpolation for image synthesis. In *Proc. SIGGRAPH 93*, pages 279–288, 1993.

[CWS95] R. Chellappa, C. L. Wilson, and S. Sirohey. Human and machine recognition of faces: A survey. *Proc. IEEE*, 83(5):705–740, 1995.

[DNvGK97] Kristin J. Dana, Shree K. Nayar, Bram van Ginneken, and Jan J. Koenderink. Reflectance and texture of real-world surfaces. In *Proc. Computer Vision and Pattern Recognition Conf.*, pages 151–157, 1997.

[DP91] Trevor Darrell and Alex Pentland. Robust estimation of a multi-layered motion representation. In *Proc. IEEE Workshop on Visual Motion*, pages 173–178, 1991.

[DP95] Trevor Darrell and Alex P. Pentland. Cooperative robust estimation using layers of support. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 17(5):474–487, 1995.

[DTM96] Paul E. Debevec, Camillo J. Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In *Proc. SIGGRAPH 96*, pages 11–20, 1996.

[DZLF94] R. Deriche, Z. Zhang, Q.-T. Luong, and O. Faugeras. Robust recovery of the epipolar geometry for an uncalibrated stereo rig. In *Proc. European Conf. on Computer Vision*, pages 567–576, 1994.

[Fau93] Olivier Faugeras. *Three-Dimensional Computer Vision, A Geometric Viewpoint*. MIT Press, Cambridge, MA, 1993.

[FM95] Olivier Faugeras and Mourrain. On the geometry and algebra of the point and line correspondences between N images. In *Proc. Fifth Int. Conf. on Computer Vision*, pages 951–956, 1995.

[Fre92] John E. Freund. *Mathematical Statistics*. Prentice Hall, Englewood Cliffs, NJ, 1992.

[FS75] R. W. Floyd and L. Steinberg. An adaptive algorithm for spatial gray scale. In *SID 75, Int. Symp. Dig. Tech. Papers*, page 36. 1975.

[GGSC96] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. The lumigraph. In *Proc. SIGGRAPH 96*, pages 43–54, 1996.

[GLY92] Davi Geiger, Bruce Landendorf, and Alan Yuille. Occlusions and binocular stereo. In *Proc. European Conf. on Computer Vision*, pages 425–433, 1992.

[Gre86] Ned Greene. Environment mapping and other applications of world projections. *IEEE Computer Graphics and Applications*, 6(11):21–29, November 1986.

[HA95] Anders Heyden and Kalle Astrom. A canonical framework for sequences of images. In *Proc. IEEE Workshop on Representation of Visual Scenes*, pages 45–52, 1995.

[Har94] Richard I. Hartley. Projective reconstruction and invariants from multiple images. *IEEE Trans. Pattern Analysis and Machine Intell.*, 16(10):1036–1041, 1994.

[Har95] Richard I. Hartley. In defence of the 8-point algorithm. In *Proc. Fifth Int. Conf. on Computer Vision*, pages 1064–1070, 1995.

[Hec89] Paul S. Heckbert. Fundamentals of texture mapping and image warping. Master's thesis, University of California, Dept. CS, Berkeley, CA, May 1989.

[HiAA97] Youichi Horry, Ken ichi Anjyo, and Kiyoshi Arai. Tour into the picture: Using a spidery mesh interface to make animation from a single image. In *Proc. SIGGRAPH 97*, pages 225–232, 1997.

[Hil83] Ellen Catherine Hildreth. *The Measurement of Visual Motion.* MIT Press, Cambridge, MA, 1983.

[HSW92] Glenn E. Healey, Steven A. Shafer, and Lawrence B. Wolff, editors. *Physics-based vision: Principles and Practice, COLOR*, pages 205–299. Jones and Bartlett, Boston, MA, 1992.

[IAH95] Michal Irani, P. Anandan, and Steve Hsu. Mosaic based representations of video sequences and their applications. In *Proc. Fifth Int. Conf. on Computer Vision*, pages 605–611, 1995.

[Inf97] Infinite Pictures, Inc. SmoothMove, version 1.0, 1997.

[Int97] Interactive Pictures Corporation, Inc. IPIX, version 1.0, 1997.

[KAI$^+$95] Rakesh Kumar, P. Anandan, Michal Irani, James Bergen, and Keith Hanna. Representation of scenes from collections of images. In *Proc. IEEE Workshop on Representation of Visual Scenes*, pages 10–17, 1995.

[KD95a] Kiriakos N. Kutulakos and Charles R. Dyer. Affine surface reconstruction by purposive viewpoint control. In *Proc. Fifth Int. Conf. on Computer Vision*, pages 894–901, 1995.

[KD95b] Kiriakos N. Kutulakos and Charles R. Dyer. Global surface reconstruction by purposive control of observer motion. *Artificial Intelligence*, 78:147–177, 1995.

[KNR95] Takeo Kanade, P. J. Narayanan, and Peter W. Rander. Virtualized reality: Concepts and early results. In *Proc. IEEE Workshop on Representation of Visual Scenes*, pages 69–76, 1995.

[KRN97] Takeo Kanade, Peter Rander, and P. J. Narayanan. Virtualized reality: Constructing virtual worlds from real scenes. *IEEE Multimedia*, 4(1):34–46, 1997.

[KS96] Sing Bing Kang and Richard Szeliski. 3-D scene data recovery using omnidirectional multibaseline stereo. In *Proc. Computer Vision and Pattern Recognition Conf.*, pages 364–370, 1996.

[KTOT95] Akihiro Katayama, Koichiro Tanaka, Takahiro Oshino, and Hideyuki Tamura. A viewpoint dependent stereoscopic display using interpolation of multi-viewpoint images. In *Proc. SPIE Vol. 2409A*, pages 21–30, 1995.

[Kut97] Kiriakos N. Kutulakos. Shape from the light field boundary. In *Proc. Computer Vision and Pattern Recognition Conf.*, pages 53–59, 1997.

[KvD76] Jan J. Koenderink and Andrea J. van Doorn. The singularities of the visual mapping. *Biological Cybernetics*, 24:51–59, 1976.

[KvD79] Jan J. Koenderink and Andrea J. van Doorn. The internal representation of solid shape with respect to vision. *Biological Cybernetics*, 32:211–216, 1979.

[Las87] John Lasseter. Principles of traditional animation applied to 3D computer animation. In *Proc. SIGGRAPH 87*, pages 35–44, 1987.

[Lau95] Aldo Laurentini. How far 3D shapes can be understood from 2D silhouettes. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 17(2):188–195, 1995.

[Lau97] Aldo Laurentini. How many 2D silhouettes does it take to reconstruct a 3D object. *Computer Vision and Image Understanding*, 67(1):81–87, 1997.

[LCSW92] Seung-Yong Lee, Kyung-Yong Chwa, Sung Yong Shin, and George Wolberg. Image metamorphosis using snakes and free-form deformations. In *Proc. SIGGRAPH 92*, pages 439–448, 1992.

[LF94] Stephane Laveau and Olivier Faugeras. 3-D scene representation as a collection of images. In *Proc. Int. Conf. on Pattern Recognition*, pages 689–691, 1994.

[LF96] Quang-Tuan Luong and Olivier Faugeras. The fundamental matrix: Theory, algorithms, and stability analysis. *Int. J. of Computer Vision*, 17(1):43–75, 1996.

[LH81] H. C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293:133–135, 1981.

[LH91] David Laur and Pat Hanrahan. Hierarchical splatting: A progressive refinement algorithm for volume rendering. In *Proc. SIGGRAPH 91*, 1991.

[LH96] Marc Levoy and Pat Hanrahan. Light field rendering. In *Proc. SIGGRAPH 96*, 1996.

[Lip80] A. Lippman. Movie-maps: an application of the optical videodisc to computer graphics. In *Proc. SIGGRAPH 80*, pages 32–42, 1980.

[Liv97] Live Picture, Inc. RealSpace Viewer, version 2.0, 1997.

[LS97] Jed Lengyel and John Snyder. Rendering with coherent layers. In *Proc. SIGGRAPH 97*, pages 233–242, 1997.

[LV94] Q-.T. Luong and T. Viéville. Canonic representations for the geometries of multiple projective views. In *Proc. Third European Conf. on Computer Vision*, pages 589–597, 1994.

[MA91] W. N. Martin and J. K. Aggarwal. Volumetric description of objects from multiple views. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 5(2):150–158, 1991.

[Mar82] David Marr. *Vision*. W. H. Freeman Co., San Francisco, CA, 1982.

[MB95a] Leonard McMillan and Gary Bishop. Head-tracked stereoscopic display using image warping. In *Proc. SPIE Vol. 2409A*, pages 21–30, 1995.

[MB95b] Leonard McMillan and Gary Bishop. Plenoptic modeling. In *Proc. SIGGRAPH 95*, pages 39–46, 1995.

[MKKJ96] Saied Moezzi, Arun Katkere, Don Y. Kuramura, and Ramesh Jain. Reality modeling and visualization from multiple video sequences. *IEEE Computer Graphics and Applications*, 16(6):58–63, 1996.

[MN90] Hiroshi Murase and Shree K. Nayar. Visual learning and recognition of 3-D objects from appearance. *Int. J. of Computer Vision*, 14(1):171–183, 1990.

[MP95] Steve Mann and Rosalind Picard. Video orbits of the projective group; A new perspective on image mosaicing. A.I. Memo No. 338, MIT Media Lab, Cambridge, MA, 1995.

[MP97] Steve Mann and Rosalind Picard. Video orbits of the projective group: A simple approach to featureless estimation of parameters. *IEEE Trans. on Image Processing*, 6(9), 1997.

[MTG97] Saied Moezzi, Li-Cheng Tai, and Philippe Gerard. Virtual view generation for 3d digital video. *IEEE Multimedia*, 4(1):18–26, 1997.

[NMSO96] Yuichi Nakamura, Tomohiko Matsuura, Kiyohide Satoh, and Yuichi Ohta. Occlusion detectable stereo–occlusion patterns in camera matrix. In *Proc. Computer Vision and Pattern Recognition Conf.*, pages 371–378, 1996.

[OK85a] Yuichi Ohta and Takeo Kanade. Stereo by intra- and inter-scanline search using dynamic programming. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 7(2):139–154, 1985.

[OK85b] Masatoshi Okutomi and Takeo Kanade. A multiple-baseline stereo. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 15(4):353–363, 1985.

[OLC93] Maximilian Ott, John P. Lewis, and Ingemar Cox. Teleconferencing eye contact using a virtual camera. In *Proc. INTERCHI '93*, pages 109–110, 1993.

[PB92] Tomaso Poggio and Roberto Brunelli. A novel approach to graphics. A.I. Memo No. 1354, M.I.T., Cambridge, MA, February 1992.

[Per96] Pietro Perona. Recognition of planar object classes. In *Proc. Computer Vision and Pattern Recognition Conf.*, pages 223–230, 1996.

[PMS94] Alex Pentland, Baback Moghaddam, and Thad Starner. View-based and modular eigenspaces for face recognition. In *Proc. Computer Vision and Pattern Recognition Conf.*, pages 84–91, 1994.

[PTK85] Tomaso Poggio, Vincent Torre, and Christof Koch. Computational vision and regularization theory. *Nature*, 317:314–319, 1985.

[Rob97] Luc Robert. Realistic scene models from image sequences. In *Proc. Imagina 97 Conf.*, pages 8–13, Monaco, 1997.

[RP94] Matthew Regan and Ronald Post. Priority rendering with a virtual reality address recalculation pipeline. In *Proc. SIGGRAPH 94*, pages 155–162, 1994.

[RZFH95] Luc Robert, Cyril Zeller, Olivier Faugeras, and Martial Hébert. Applications of non-metric vision to some visually guided robotics tasks. Technical Report 2584, INRIA, Sophia-Antipolis, France, June 1995.

[Sam84] H. Samet. The quadtree and related hierarchical data structures. *ACM Computing Surveys*, 16:187–260, 1984.

[SCG97] Peter-Pike Sloan, Michael F. Cohen, and Steven J. Gortler. Time critical lumigraph rendering. In *Proc. Symposium on Interactive 3D Graphics*, pages 17–23, 1997.

[Sch96] Daniel Scharstein. Stereo vision for view synthesis. In *Proc. Computer Vision and Pattern Recognition Conf.*, pages 852–858, 1996.

[SD88] C. V. Stewart and C. R. Dyer. The trinocular general support algorithm: a three-camera stereo algorithm for overcoming binocular matching errors. In *Proc. Second Int. Conf. on Computer Vision*, pages 134–138, 1988.

[SD95a] Steven M. Seitz and Charles R. Dyer. Complete structure from four point correspondences. In *Proc. Fifth Int. Conf. on Computer Vision*, pages 330–337, 1995.

[SD95b] Steven M. Seitz and Charles R. Dyer. Physically-valid view synthesis by image interpolation. In *Proc. IEEE Workshop on Representation of Visual Scenes*, pages 18–25, 1995.

[SD96a] Steven M. Seitz and Charles R. Dyer. Scene appearance representation by perspective view synthesis. Technical Report CS-TR-1298, University of Wisconsin, Madison, WI, May 1996.

[SD96b] Steven M. Seitz and Charles R. Dyer. Toward image-based scene representation using view morphing. In *Proc. Int. Conf. on Pattern Recognition*, pages 84–89, 1996.

[SD96c] Steven M. Seitz and Charles R. Dyer. View morphing. In *Proc. SIGGRAPH 96*, pages 21–30, 1996.

[SD97a] Steven M. Seitz and Charles R. Dyer. Photorealistic scene reconstruction by voxel coloring. In *Proc. Image Understanding Workshop*, pages 1067–1073, 1997.

[SD97b] Steven M. Seitz and Charles R. Dyer. Photorealistic scene reconstruction by voxel coloring. In *Proc. Computer Vision and Pattern Recognition Conf.*, pages 1067–1073, 1997.

[SD97c] Steven M. Seitz and Charles R. Dyer. View-invariant analysis of cyclic motion. *Int. J. of Computer Vision*, 25(3), 1997. To appear.

[SD97d] Steven M. Seitz and Charles R. Dyer. View morphing: Uniquely predicting scene appearance from basis images. In *Proc. Image Understanding Workshop*, pages 881–887, 1997.

[Sei97] Steven M. Seitz. Bringing photographs to life with view morphing. In *Proc. Imagina 97 Conf.*, pages 153–158, Monaco, 1997.

[SF95] W. Brent Seales and Olivier D. Faugeras. Building three-dimensional object models from image sequences. *CVGIP: Image Understanding*, 3(61):308–324, 1995.

[Sha94] Amnon Shashua. Projective structure from uncalibrated images: Structure from motion and recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 16(8):778–790, 1994.

[Sho85] Ken Shoemake. Animating rotation with quaternion curves. In *Proc. SIGGRAPH 85*, pages 245–254, 1985.

[SI92] A. Samal and P. Iyengar. Automatic recognition and analysis of human faces and facial expressions: A survey. *Pattern Recognition*, 25(5):65–77, 1992.

[SK95] Richard Szeliski and Sing Bing Kang. Direct methods for visual scene reconstruction. In *Proc. IEEE Workshop on Representation of Visual Scenes*, 1995. To appear.

[SK98] Steven M. Seitz and Kiriakos N. Kutulakos. Plenoptic image editing. In *Proc. Sixth Int. Conf. on Computer Vision*, 1998. To appear.

[SS97] Richard Szeliski and Heung-Yeung Shum. Creating full view panoramic image mosaics and environment maps. In *Proc. SIGGRAPH 97*, pages 251–258, 1997.

[SWI97] Yoichi Sato, Mark D. Wheeler, and Katsushi Ikeuchi. Object shape and reflectance modeling from observation. In *Proc. SIGGRAPH 97*, pages 379–387, 1997.

[SY97] W. Brent Seales and Cheng Jiun Yuan. Recognition using morphing. In preparation, 1997.

[SZB95] Larry S. Shapiro, Andrew Zisserman, and Michael Brady. 3D motion recovery via affine epipolar geometry. *Int. J. of Computer Vision*, 16:147–182, 1995.

[Sze93] Richard Szeliski. Rapid octree construction from image sequences. *Computer Vision, Graphics, and Image Processing: Image Understanding*, 1(58):23–32, 1993.

[Sze96] Richard Szeliski. Video mosaics for virtual environments. *IEEE Computer Graphics and Applications*, 16(2):22–30, 1996.

[TK92] Carlo Tomasi and Takeo Kanade. Shape and motion from image streams under orthography: A factorization method. *Int. J. of Computer Vision*, 9(2):137–154, 1992.

[TK96] Jay Torborg and James T. Kajiya. Talisman: Commodity realtime 3D graphics for the PC. In *Proc. SIGGRAPH 96*, pages 353–363, 1996.

[TL94] Greg Turk and Marc Levoy. Zippered polygon meshes from range images. In *Proc. SIGGRAPH 94*, pages 311–318, 1994.

[TP91] Matthew Turk and Alex Pentland. Eigenfaces for recognition. *J. of Cognitive Neuroscience*, 3(1):71–86, 1991.

[TP94] Sebastian Toelg and Tomaso Poggio. Towards an example-based image compression architecture for video conferencing. A.I. Memo No. 1494, M.I.T., Cambridge, MA, June 1994.

[Tri95] Bill Triggs. Matching constraints and the joint image. In *Proc. 5th Int. Conf. on Computer Vision*, pages 338–343, 1995.

[Tsa87] R. Y. Tsai. A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf cameras and lenses. *IEEE Trans. Robotics and Automation*, 3(4):323–344, 1987.

[UB91] Shimon Ullman and Ronen Basri. Recognition by linear combinations of models. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 13(10):992–1006, 1991.

[VP89] Alessandro Verri and Tomaso Poggio. Motion field and optical flow: Qualitative properties. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 11(5):490–498, 1989.

[VP96] Thomas Vetter and Tomaso Poggio. Image synthesis from a single example image. In *Proc. European Conf. on Computer Vision*, pages 653–659, 1996.

[WA94] John Y. A. Wang and Edward H. Adelson. Representing moving images with layers. *IEEE Trans. Image Processing*, 3(5):625–638, 1994.

[Wal33] Walt Disney Productions. Three little pigs. film, 1933.

[Wal40a] Walt Disney Productions. Fantasia. film, 1940.

[Wal40b] Walt Disney Productions. Pinocchio. film, 1940.

[Wal92] Walt Disney Productions. Aladdin. film, 1992.

[War92] Gregory J. Ward. Measuring and modeling anisotropic reflection. In *Proc. SIGGRAPH 92*, pages 265–272, 1992.

[Wes90] Lee Westover. Footprint evaluation for volume rendering. In *Proc. SIGGRAPH 90*, pages 367–376, 1990.

[WFH$^+$97] Daniel N. Wood, Adam Finkelstein, John F. Hughes, Craig E. Thayer, and David H. Salesin. Multiperspective panoramas for cel animation. In *Proc. SIGGRAPH 97*, pages 243–250, 1997.

[WHH95] Tomas Werner, Roger David Hersch, and Vaclav Hlavac. Rendering real-world objects using view interpolation. In *Proc. Fifth Int. Conf. on Computer Vision*, pages 957–962, 1995.

[Wol90] George Wolberg. *Digital Image Warping*. IEEE Computer Society Press, Los Alamitos, CA, 1990.

[ZW96] C. Lawrence Zitnick and Jon A. Webb. Multi-baseline stereo using surface extraction. Technical Report CMU-CS-96-196, Carnegie Mellon University, Pittsburgh, PA, November 1996.

# Appendix A

# Computing Prewarps

In this section we consider the problem of computing the prewarp transformations from a set of point correspondences. This procedure is needed to apply the view morphing algorithm in situations when the camera configurations are not known.

As described in Section 3.9.2, prewarped images $\hat{\mathcal{I}}_0$ and $\hat{\mathcal{I}}_1$ satisfy the *scanline property*, which dictates that corresponding points in the two images appear in the same scanline. In fact, achieving this scanline property is both necessary and sufficient to ensure that two images have been properly prewarped. In this section we first demonstrate the necessity and sufficiency of the scanline property. Then we present an algorithm which uses the scanline property to compute the prewarp transformations from a set of point correspondences.

## A.1 The Fundamental Matrix

In order to use the three-step algorithm presented in Section 3.9.2, we must find a way to prewarp the images without knowing the projection matrices. Towards this end, we first consider what form the *fundamental matrix* takes when the views satisfy the scanline property and then prewarp the views in such a way so that the fundamental matrix achieves this form.

The fundamental matrix of two images $\mathcal{I}_0$ and $\mathcal{I}_1$ is the $3 \times 3$, rank-two matrix $\mathbf{F}$ satisfying the following relation [LH81, LF96]

$$\mathbf{p}_1^T \mathbf{F} \mathbf{p}_0 = 0$$

for any pair of points $\mathbf{p}_0 \in \mathcal{I}_0$ and $\mathbf{p}_1 \in \mathcal{I}_1$ corresponding to the same scene point. $\mathbf{F}$ is defined up to a scale factor and can be computed from the images themselves when at least 8 point correspondences are known (see [LF96, Har95, SZB95] for methods of computing $\mathbf{F}$ from point correspondences).

Let $\mathcal{I}_0$ and $\mathcal{I}_1$ be two views with projection matrices $\mathbf{\Pi}_0 = [\mathbf{I} \mid \mathbf{0}]$ and $\mathbf{\Pi}_1 = [\mathbf{H}_1 \mid -\mathbf{H}_1\mathbf{C}_1]$. Without loss of generality, we have assumed the first camera to be centered at the world origin and have set the world $X$ and $Y$ axes to coincide with the image coordinate axes of $\mathcal{I}_0$. The epipoles are the projections of $\mathbf{C}_1$ into $\mathcal{I}_0$ and $\mathbf{C}_0$ into $\mathcal{I}_1$:

$$\tilde{\mathbf{e}}_0 = \mathbf{C}_1 \qquad (A.16)$$

$$\tilde{\mathbf{e}}_1 = -\mathbf{H}_1\mathbf{C}_1 \qquad (A.17)$$

Given a vector $\mathbf{p} = [x\ y\ z]^T$, we use the notation

$$[\mathbf{p}]_\times = \begin{bmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{bmatrix}$$

Following [LF96], the fundamental matrix may be expressed as

$$\mathbf{F} = [\mathbf{e}_1]_\times \mathbf{H}_1$$

Recall that prewarped images satisfy the scanline property and represent views with projection matrices $\hat{\mathbf{\Pi}}_0 = [\mathbf{I} \mid -\mathbf{C}_0]$ and $\hat{\mathbf{\Pi}}_1 = [\mathbf{I} \mid -\mathbf{C}_1]$, where $\mathbf{I}$ is the $3 \times 3$ identity matrix. Therefore, $\mathbf{H}_1 = \mathbf{I}$ and $\tilde{\mathbf{e}}_1 = [e_x\ 0\ 0]^T$ for some constant $e_x$. Consequently, the fundamental matrix is given, up to a scale constant, by

$$\hat{\mathbf{F}} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \tag{A.18}$$

Conversely, suppose $\mathbf{F}$ is given by Eq. (A.18) and the unknown projection matrices are $\mathbf{\Pi}_0 = [\mathbf{I} \mid \mathbf{0}]$ and $\mathbf{\Pi}_1 = [\mathbf{H}_1 \mid -\mathbf{H}_1\mathbf{C}_1]$. The epipoles are such that $\hat{\mathbf{F}}\tilde{\mathbf{e}}_0 = \mathbf{e}_1^T\hat{\mathbf{F}} = \mathbf{0}$ [LF96]. In particular, $\tilde{\mathbf{e}}_0$ must have the form

$$\tilde{\mathbf{e}}_0 = [e_x\ 0\ 0]^T$$

for some unknown constant $e_x$. From Eq. (A.16) it follows that $\mathbf{C}_1 = [e_x\ 0\ 0]^T$. To make the views parallel, it therefore suffices to transform $\mathcal{I}_1$ to $\mathbf{H}_1^{-1}\mathcal{I}_1$. This image transformation induces a corresponding change in the fundamental matrix, to $\mathbf{H}_1^T\hat{\mathbf{F}}$. By the preceding argument, however, the prewarped fundamental matrix is fixed, up to scalar multiple, by Eq. (A.18). We therefore have the following constraint on $\mathbf{H}_1$:

$$\mathbf{H}_1^T\hat{\mathbf{F}} = \hat{\mathbf{F}}$$

It follows that $\mathbf{H}_1$ has the following structure:

$$\mathbf{H}_1 = \begin{bmatrix} a & b & c \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

for unknown constants $a$, $b$, and $c$. Consequently, the third rows of $\mathbf{\Pi}_0$ and $\mathbf{\Pi}_1$ are equal so the views are parallel. Also note that the second rows are equal, thus ensuring that the scanline property is satisfied.

In summary, Eq. (A.18) provides a necessary and sufficient condition for testing whether two views are parallel and satisfy the scanline property from their fundamental matrix. In order to make use of this test, we seek a pair of homographies $\mathbf{H}_0$ and $\mathbf{H}_1$ such that the prewarped images $\hat{\mathcal{I}}_0 = \mathbf{H}_0^{-1}\mathcal{I}_0$

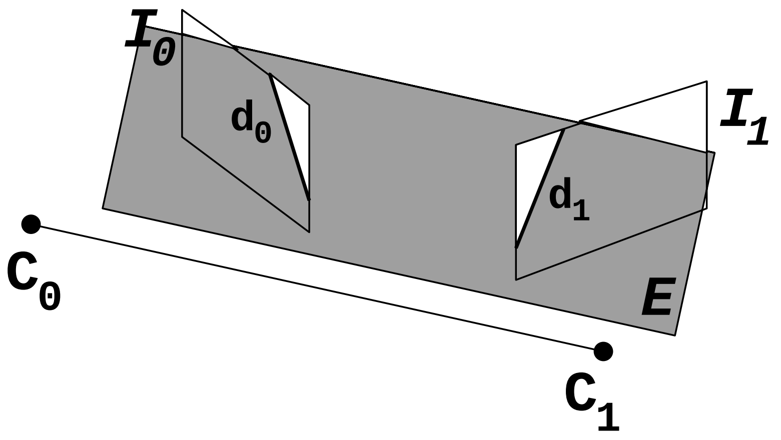


Figure A.1: A pair of views can be made coincident with a plane $E$ by rotating the image planes $\mathcal{I}_i$ about their lines of intersection $\mathbf{d}_i$ with $E$. $\mathcal{I}_i$ can be made parallel to $E$ by rotating the image plane about any line parallel to $\mathbf{d}_i$.

and $\hat{\mathcal{I}}_1 = \mathbf{H}_1^{-1}\mathcal{I}_1$ have the fundamental matrix given by Eq. (A.18). In terms of $\mathbf{F}$ the condition on $\mathbf{H}_0$ and $\mathbf{H}_1$ is

$$\mathbf{H}_1{}^T\mathbf{F}\mathbf{H}_0 = \hat{\mathbf{F}} \tag{A.19}$$

## A.2 Choosing the Homographies

There is in fact a range of homographies $\mathbf{H}_0$ and $\mathbf{H}_1$ satisfying Eq. (A.19), corresponding to different choices of the prewarp plane. As in the calibrated case, the particular choice of homographies is not important, except in regard to sampling considerations (see Section 3.9). Here we describe one method that applies a 3D rotation to make the image planes parallel, followed by a 2D affine transformation to align corresponding scanlines. A nice property of this method is that the prewarp process reduces to the technique in Appendix B when the images are orthographic.

### A.2.1 Aligning the Image Planes

The idea behind the technique is depicted in Fig. A.1. Let $E$ be a plane parallel to $\overline{\mathbf{C}_0\mathbf{C}_1}$. Two views may be made parallel by reprojecting them onto $E$. One way of performing the reprojection is to rotate each image plane $\mathcal{I}_i$ about the line $\mathbf{d}_i$ given by the intersection of $E$ with $\mathcal{I}_i$. Alternatively, rotating $\mathcal{I}_i$ about any line parallel to $\mathbf{d}_i$ is sufficient to align the image planes so that they are both parallel to $E$.

The first step of the prewarp procedure is to choose $E$. This is done indirectly, by selecting its intersection $\mathbf{d}_0$ with $\mathcal{I}_0$. For convenience, we fix this line to pass through the image origin and represent it as the homogeneous unit row vector $\tilde{\mathbf{d}}_0 = [-d_0^y \; d_0^x \; 0]$. Points $\mathbf{p}$ on this line have the form $\mathbf{p} = [sd_0^x \; sd_0^y \; 1]^T$ and satisfy the equation

$$\tilde{\mathbf{d}}_0\mathbf{p} = 0$$

Recall that the epipole $\mathbf{e}_0$ represents the projection of $\mathbf{C}_1$ into $\mathcal{I}_0$, i.e., the intersection of $\mathbf{C}_0\mathbf{C}_1$ with $\mathcal{I}_0$. An image plane parallel to $\overline{\mathbf{C}_0\mathbf{C}_1}$ therefore has epipole at infinity. Therefore, we seek a rotation $\mathbf{R}_{\theta_0}^{\mathbf{d}_0}$ such that the new epipole $\hat{\mathbf{e}}_0 = \mathbf{R}_{\theta_0}^{\mathbf{d}_0}\tilde{\mathbf{e}}_0$ has the form $\hat{\mathbf{e}}_0 = [\hat{e}_0^x\ \hat{e}_0^y\ 0]^T$.

The rotation matrix is given by

$$\mathbf{R}_{\theta_0}^{\mathbf{d}_0} = \begin{bmatrix} (d_0^x)^2 + (1-(d_0^x)^2)cos\ \theta_0 & d_0^x d_0^y(1-cos\ \theta_0) & d_0^y sin\ \theta_0 \\ d_0^x d_0^y(1-cos\ \theta_0) & (d_0^y)^2 + (1-(d_0^y)^2)cos\ \theta_0 & -d_0^x sin\ \theta_0 \\ -d_0^y sin\ \theta_0 & d_0^x sin\ \theta_0 & cos\ \theta_0 \end{bmatrix} \tag{A.20}$$

From Eq. (A.20), the desired angle of rotation $\theta_0$ is determined to be

$$\theta_0 = tan^{-1}(\frac{e_z}{d_0^y e_0^x - d_0^x e_0^y}) \tag{A.21}$$

An analogous rotation is performed on $\mathcal{I}_1$. To determine the axis of rotation $\mathbf{d}_1$, let $E'$ be the epipolar plane parallel to $E$. $E'$ intersects $\mathcal{I}_i$ in an epipolar line $\mathbf{l}_i$ parallel to $\mathbf{d}_i$. Because they are parallel, $\mathbf{l}_0$ and $\mathbf{d}_0$ intersect at the ideal (infinite) point $\tilde{\mathbf{i}}_0 = [d_0^x\ d_0^y\ 0]$. We can compute the correspondence between conjugate epipolar lines $\mathbf{l}_0$ and $\mathbf{l}_1$ using the fundamental matrix as follows [LF96]:

$$\tilde{\mathbf{l}}_1 = \mathbf{F}\tilde{\mathbf{i}}_0 \tag{A.22}$$

$\mathcal{I}_1$ may be made parallel to $E'$ (and hence to $E$) by rotating about any line $\mathbf{d}_1$ parallel to $\mathbf{l}_1$. It is convenient to choose $\mathbf{d}_1$ so that, like $\mathbf{d}_0$, it passes through the image origin. Accordingly, if $[x\ y\ z]^T = \mathbf{F}[d_0^x\ d_0^y\ 0]^T$ then $\tilde{\mathbf{d}}_1 = \alpha[x\ y\ 0]^T$, i.e., $d_1^x = \alpha y$ and $d_1^y = -\alpha x$, where $\alpha = \frac{1}{\sqrt{x^2+y^2}}$.

### A.2.2 Aligning the Scanlines

Applying $\mathbf{R}_{\theta_0}^{\mathbf{d}_0}$ to $\mathcal{I}_0$ and $\mathbf{R}_{\theta_1}^{\mathbf{d}_1}$ to $\mathcal{I}_1$ makes the two image planes parallel. More specifically, $\mathbf{R}_{\theta_0}^{\mathbf{d}_0}\mathcal{I}_0$ is coincident with $E$ and $\mathbf{R}_{\theta_1}^{\mathbf{d}_1}\mathcal{I}_1$ is parallel to $E$. Although this is technically sufficient for prewarping, it is useful to add an additional affine warp to align the scanlines. This simplifies the morph step to a scanline interpolation and also avoids bottleneck problems that arise as a result of image plane rotations [Wol90].

The next step is to rotate the images so that epipolar lines are horizontal. The new epipoles are $[\hat{e}_i^x\ \hat{e}_i^y\ 0]^T = \mathbf{R}_{\theta_i}^{\mathbf{d}_i}\mathbf{e}_i$. These image plane ($z$ axis) rotations are given by

$$\begin{aligned} \phi_i &= -tan^{-1}(\hat{e}_i^y/\hat{e}_i^x) \\ \mathbf{R}_{\phi_i} &= \begin{bmatrix} cos\ \phi_0 & -sin\ \phi_0 & 0 \\ sin\ \phi_0 & cos\ \phi_0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{aligned}$$

After applying these image plane rotations, the fundamental matrix has the form (up to a scale factor)

$$\tilde{\mathbf{F}} = \mathbf{R}_{\phi_1}\mathbf{R}_{\theta_1}^{\mathbf{d}_1}\mathbf{F}\mathbf{R}_{-\theta_0}^{\mathbf{d}_0}\mathbf{R}_{-\phi_0} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & a \\ 0 & 1 & b \end{bmatrix} \tag{A.23}$$

Applying $\mathbf{R}_{\phi_i}\mathbf{R}_{\theta_i}^{\mathbf{d}_i}$ to $\mathcal{I}_i$ results in a pair of images with horizontal epipolar lines. It is possible, however, that the epipolar lines appear in opposite top-to-bottom order in the two images, in which case an additional $180°$ rotation is needed. This condition may be determined by checking the signs of $a$ and $b$ in Eq. (A.23). If $ab > 0$ then $\phi_1$ should be incremented by $\pi$.

Finally, to get $\tilde{\mathbf{F}}$ into the form of Eq. (A.18), the second image is vertically scaled and translated by the matrix

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -a & -b \\ 0 & 0 & 1 \end{bmatrix}$$

It is easily checked that $\mathbf{T}^{-1^T}\tilde{\mathbf{F}} = \hat{\mathbf{F}}$. In summary, the prewarping transforms $\mathbf{H}_0^{-1}$ and $\mathbf{H}_1^{-1}$ are

$$\begin{aligned} \mathbf{H}_0^{-1} &= \mathbf{R}_{\phi_0}\mathbf{R}_{\theta_0}^{\mathbf{d}_0} \\ \mathbf{H}_1^{-1} &= \mathbf{T}\mathbf{R}_{\phi_1}\mathbf{R}_{\theta_1}^{\mathbf{d}_1} \end{aligned}$$

### A.2.3 Infinite Epipoles

One case that requires special attention is the scenario in which the original image planes are both parallel to $\overline{\mathbf{C}_0\mathbf{C}_1}$ but are not parallel to a common epipolar plane. In this case, the epipoles are already at infinity so $\mathbf{R}_{\theta_0}^{\mathbf{d}_0}$ and $\mathbf{R}_{\theta_1}^{\mathbf{d}_1}$ should be omitted. Accordingly, the new fundamental matrix has the form

$$\tilde{\mathbf{F}} = \mathbf{R}_{\phi_1}\mathbf{F}\mathbf{R}_{-\phi_0} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & c & d \\ 0 & e & f \end{bmatrix} \tag{A.24}$$

To make the image planes parallel and convert the fundamental matrix to the form of Eq. (A.23), an additional $x$-axis rotation of $\mathcal{I}_1$ is needed:

$$\begin{aligned} \gamma &= tan^{-1}\frac{c}{e} \\ \mathbf{R}_\gamma &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\,\gamma & -sin\,\gamma \\ 0 & sin\,\gamma & cos\,\gamma \end{bmatrix} \end{aligned}$$

In this case, $\mathbf{T}$ should be computed from

$$\tilde{\mathbf{F}} = \mathbf{R}_\gamma \mathbf{R}_{\phi_1} \mathbf{F} \mathbf{R}_{-\phi_0}$$

and

$$\begin{aligned} \mathbf{H}_0^{-1} &= \mathbf{R}_{\phi_0} \\ \mathbf{H}_1^{-1} &= \mathbf{T}\mathbf{R}_\gamma \mathbf{R}_{\phi_1} \end{aligned}$$

### A.2.4 Selecting $\mathbf{d}_0$

The entire procedure is determined by selecting $\mathbf{d}_0$. In principle, any choice of $\mathbf{d}_0$ will suffice. In practice, it is useful to choose the axis of rotation so as to minimize nonlinear distortion in the prewarped images. Prewarping an image requires resampling the transformed image onto an integer grid of pixels. While linear image maps (e.g., rotations) can be efficiently resampled using space-invariant filters [Hec89], nonlinear transformations often lead to suboptimal reconstructions. It is therefore beneficial to pick $\mathbf{d}_0$ so as to minimize nonlinear effects. The nonlinear part of the prewarp transformation is due solely to $\mathbf{R}_{\theta_i}^{\mathbf{d}_i}$.

One approach is simply to minimize $|\theta_0|$. By Eq. (A.21), minimizing $|\theta_0|$ is equivalent to minimizing the following expression:

$$\left| \frac{e_z}{d_0^y e_0^x - d_0^x e_0^y} \right|$$

The optimal value of $|\theta_0|$ corresponds to $d_0^x = \alpha e_0^y$ and $d_0^y = -\alpha e_0^x$, i.e., $\tilde{\mathbf{d}}_0 = -\alpha[e_0^x \; e_0^y \; 0]$, where $\alpha = \frac{1}{\sqrt{(e_0^x)^2+(e_0^y)^2}}$.

We note that Robert et al. [RZFH95] also considered the problem of computing prewarp transformations within the context of uncalibrated stereo rectification. Working independently, they developed a similar technique of computing reprojection mappings to minimize image distortions. An advantage of the technique presented here is that the reprojection is decomposed into a number of intuitive, component warps that are individually useful. For instance, in cases where parallel planes are sufficient, the additional steps to align scanlines may be left out. This is particularly useful when *aggregate warps* are used, as described in Section 3.11.2, and prewarped images are never explicitly constructed. Alternatively, if the images are orthographic the rotations about $\mathbf{d}_i$ should be omitted, as described below.

### A.2.5 Orthographic Prewarps

Orthography provides a good approximation to perspective projection when the field of view is small and the scene depth along the light of sight is small compared to the camera distance. The advantage of the orthographic model is that it is simpler and leads to more stable computations of parameters like epipolar lines [SZB95]. When images are close to being orthographic, it is therefore advisable to use a simplified technique for computing the fundamental matrix. As described by Shapiro et. al [SZB95], the orthographic fundamental matrix has the form

$$\mathbf{F} = \begin{bmatrix} 0 & 0 & a \\ 0 & 0 & b \\ c & d & e \end{bmatrix}$$

and may be computed from 4 or more point correspondences using linear least squared techniques. It is easily seen that the epipoles are given by:

$$\begin{aligned} \tilde{\mathbf{e}}_0 &= \frac{1}{\sqrt{d^2 + c^2}}[-d\ c\ 0]^T \\ \tilde{\mathbf{e}}_1 &= \frac{1}{\sqrt{b^2 + a^2}}[-b\ a\ 0]^T \end{aligned}$$

For the orthographic case, the epipoles are already at infinity so we may leave out rotations about $\mathbf{d}_i$. The orthographic prewarps are hence given by

$$\begin{aligned} \mathbf{H}_0^{-1} &= \mathbf{R}_{\phi_0} \\ \mathbf{H}_1^{-1} &= \mathbf{T}\mathbf{R}_{\phi_1} \end{aligned}$$

# Appendix B

# Orthographic View Morphing

This appendix presents the monotonicity constraint for orthographic views and its impact for view synthesis.

Monotonicity states that the projections of any two points on the same epipolar plane appear in the same order along conjugate epipolar lines $l_0$ and $l_1$. If this property holds for all corresponding epipolar lines in the two views then we say that monotonicity holds for $\mathcal{I}_0$ and $\mathcal{I}_1$. Let $\mathbf{P}$ and $\mathbf{Q}$ be two scene points on the same epipolar plane that are visible in both images. Geometrically, the constraint dictates that the line through $\mathbf{P} - \mathbf{Q}$ may not intersect the line segment $\overline{\mathbf{N}_0\mathbf{N}_1}$ joining the tips of the two view normals.

A useful property of monotonicity is that it extends to cover a range of views in-between $V_0$ and $V_1$. We say that a third view $V_s$ is *in-between* $V_0$ and $V_1$ if its normal $\mathbf{N}_s$ intersects $\overline{\mathbf{N}_0\mathbf{N}_1}$. Because the line through $\mathbf{P}$ and $\mathbf{Q}$ intersects $\overline{\mathbf{N}_0\mathbf{N}_1}$ if and only if it intersects either $\overline{\mathbf{N}_0\mathbf{N}_s}$ or $\overline{\mathbf{N}_s\mathbf{N}_1}$, monotonicity of $V_0$ and $V_1$ implies monotonicity of $V_0$ and $V_s$ as well as $V_s$ and $V_1$. That is, any two points on an epipolar plane must appear in the same order on corresponding epipolar lines of all three images. This property, that monotonicity applies to *in-between* views, is quite powerful and is sufficient to uniquely predict the appearance of the visible scene from all viewpoints in-between $V_1$ and $V_2$. The proof of this result is identical to the perspective case, given in Section 3.5.1. Fig. B.1 illustrates the impact of the monotonicity constraint on view synthesis.

## B.1 Image Interpolation

In Section 3.7 it was argued that linear interpolation of two orthographic views produces another orthographic view. More specifically, if $V_0$ and $V_1$ are two orthographic views, then linear interpolation of points $\mathbf{p}_0 \in \mathcal{I}_0$ $\mathbf{p}_1 \in \mathcal{I}_1$

$$\mathbf{p}_s = s\mathbf{p}_0 + (1-s)\mathbf{p}_1$$

creates a new view $V_s$ with projection matrix $\mathbf{\Pi}_s$ corresponding to

$$\mathbf{\Pi}_s = s\mathbf{\Pi}_0 + (1-s)\mathbf{\Pi}_1$$

In other words, interpolation of images in 2D has a direct physical interpretation in terms of views, a connection that was recognized by Ullman and Basri [UB91] in the context of object recognition. In spite of this result, image interpolations do not account for changes in visibility and often correspond

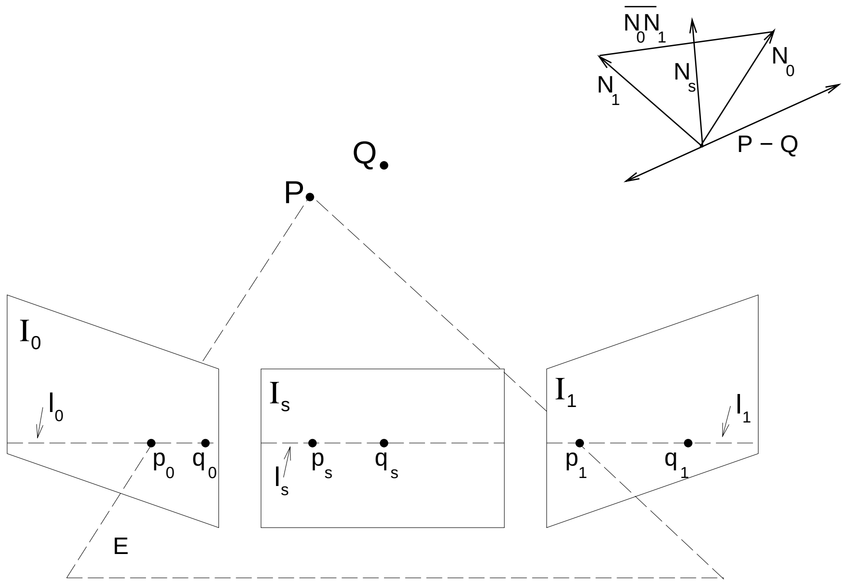
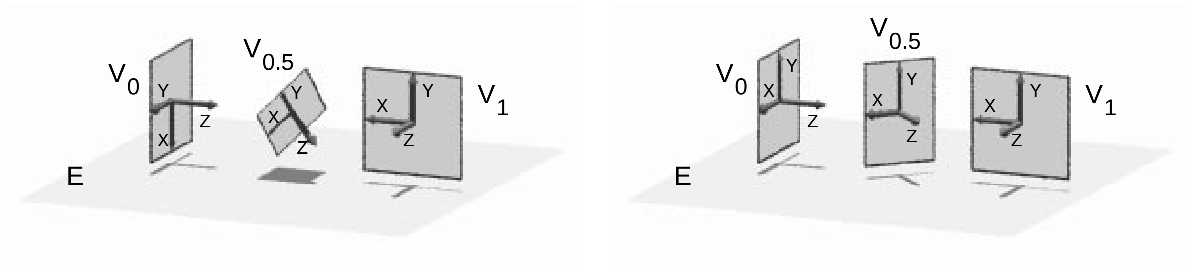


Figure B.1: Monotonic Viewing Geometry. If $\mathbf{P}$ appears to the left of $\mathbf{Q}$ in images $\mathcal{I}_0$ and $\mathcal{I}_1$ then it must also in $\mathcal{I}_s$, providing $\mathbf{N}_s$ intersects $\overline{\mathbf{N}_0\mathbf{N}_1}$. Monotonicity requires that line $\mathbf{P} - \mathbf{Q}$ does not intersect $\overline{\mathbf{N}_0\mathbf{N}_1}$.



**(a)** **(b)**

Figure B.2: Views Generated by Image Interpolation. **(a)** Interpolating the $X$ and $Y$ axes of $V_0$ and $V_1$ produces a view that is skewed and tilted with respect to the epipolar plane $E$. **(b)** Prewarping remedies the problem by aligning the view coordinate systems prior to interpolation.

to very unintuitive view interpolations. Fig. B.2a graphically depicts the interpolation of views $V_0$ and $V_1$. Although both $V_0$ and $V_1$ are normal to the epipolar plane $E$, the interpolated view $V_{0.5}$ is tilted by 45 degrees with respect to $E$. In addition, the axes of $V_0$ and $V_1$ are orthonormal, whereas the axes of $V_{0.5}$ are neither orthogonal nor of unit length. Clearly, $V_{0.5}$ does not correspond to an *in-between* view, as defined above, so monotonicity may not be preserved and correctness of the interpolated image cannot be ensured. Furthermore, there are cases where interpolation degenerates, such as when $\mathcal{I}_1$ is a 180 degree rotation of $\mathcal{I}_0$. In this case, the morph collapses to a point, with all points mapping to the origin in $\mathcal{I}_{0.5}$. In short, image interpolation of orthographic views will generally *not* produce new views within the same monotonic range. Fortunately, however, these problems can be corrected by appropriately aligning the two images before performing the interpolation (see Fig. B.2b), as described next.

## B.2 The Need for Rectification

The odd view trajectories obtained from orthographic image interpolations arise because linear interpolation of views does not amount to linear interpolation of gaze directions. Two views $V_0$ and $V_1$ each define a direction of gaze, $\mathbf{N}_0$ and $\mathbf{N}_1$. Intuitively, we might expect the gaze direction to follow the most direct path between $\mathbf{N}_0$ and $\mathbf{N}_1$ during a smooth transition between $\mathcal{I}_0$ and $\mathcal{I}_1$ . However, this is generally not the case in view interpolation, as Fig. B.2a illustrates, due to the nonlinear relationship between plane and normal transformations[1].

A morph can be made to interpolate gaze directions and to generate valid in-between views in the same monotonic range by first aligning the coordinate axes of the two views. This is accomplished by means of a simple image rectification procedure that aligns epipolar lines in the two images. The result of rectification is that corresponding points in the two rectified images will appear on the same scanline. In other words, a point $[x_0\ y\ 1]^T \in \mathcal{I}_0$ will correspond to point $[x_1\ y\ 1]^T \in \mathcal{I}_1$. Prewarp procedures sufficient to achieve this criterion are described in [SD95b] and Appendix A.

[1]Normals are transformed by the inverse transpose of the plane coordinate transformation.