To appear in Proc. SIGGRAPH 96.

View Morphing

Steven M. Seitz   Charles R. Dyer

Department of Computer Sciences
University of Wisconsin—Madison[1]

ABSTRACT

Image morphing techniques can generate compelling 2D transitions between images. However, differences in object pose or viewpoint often cause unnatural distortions in image morphs that are difficult to correct manually. Using basic principles of projective geometry, this paper introduces a simple extension to image morphing that correctly handles 3D projective camera and scene transformations. The technique, called *view morphing*, works by prewarping two images prior to computing a morph and then postwarping the interpolated images. Because no knowledge of 3D shape is required, the technique may be applied to photographs and drawings, as well as rendered scenes. The ability to synthesize changes both in viewpoint and image structure affords a wide variety of interesting 3D effects via simple image transformations.

CR Categories and Subject Descriptors: I.3.3 [Computer Graphics]: Picture/Image Generation– viewing algorithms; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism– animation; I.4.3 [Image Processing]: Enhancement– geometric correction, registration.

Additional Keywords: Morphing, image metamorphosis, view interpolation, view synthesis, image warping.

1 INTRODUCTION

Recently there has been a great deal of interest in *morphing* techniques for producing smooth transitions between images. These techniques combine 2D interpolations of shape and color to create dramatic special effects. 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 can suggest a natural transformation between objects in the 3D world. 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.

[1]1210 W. Dayton St., Madison WI 53706
Email: {seitz | dyer}@cs.wisc.edu
Web: http://www.cs.wisc.edu/~dyer/vision.html

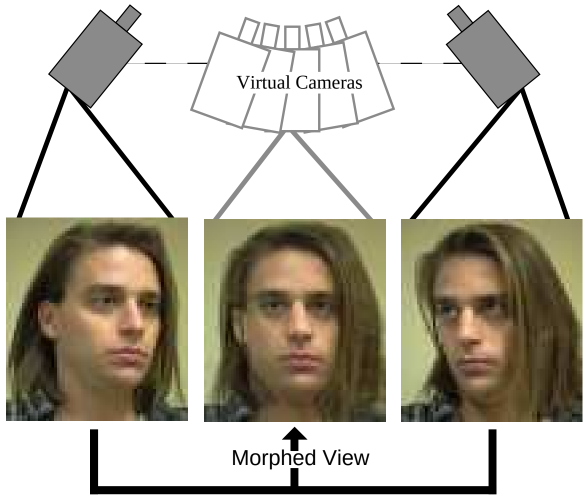

Figure 1: View morphing between two images of an object taken from two different viewpoints produces the illusion of physically moving a virtual camera.

Although current techniques enable the creation of effective image transitions, they do not *ensure* that the resulting transitions appear natural. It is entirely up to the user to evaluate a morph transition and to design the interpolation to achieve the best results. Part of the problem is that existing image morphing methods do not account for changes in viewpoint or object pose. As a result, simple 3D transformations (e.g., translations, rotations) become surprisingly difficult to convey convincingly using existing methods.

In this paper, we describe a simple extension called *view morphing* that allows current image morphing methods to easily synthesize changes in viewpoint and other 3D effects. When morphing between different views of an object or scene, the technique produces new views of the same scene, ensuring a realistic image transition. The effect can be described by what you would see if you physically moved the object (or the camera) between its configurations in the two images and filmed the transition, as shown in Fig. 1. More generally, the approach can synthesize 3D *projective transformations* of objects, a class including 3D rotations, translations, shears, and tapering deformations, by operating entirely on images (no 3D shape information is required). Because view morphing employs existing image morphing techniques as an intermediate step, it may also be used to interpolate between different views of *different* 3D objects, combining image morphing's capacity for dramatic shape transformations with view morphing's ability to achieve changes in viewpoint. The result is a simultaneous interpolation of shape, color, and pose, giving rise to image transitions that appear strikingly 3D.

View morphing works by prewarping two images, comput-

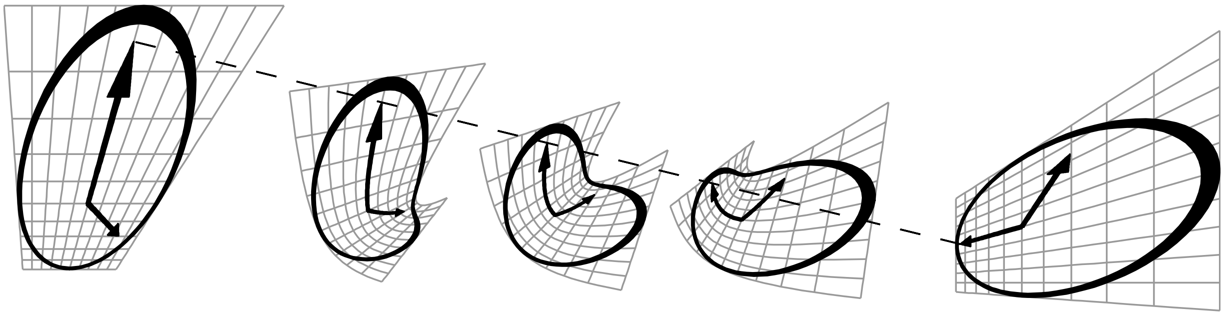

Figure 2: A Shape-Distorting Morph. Linearly interpolating two perspective views of a clock (far left and far right) causes a geometric bending effect in the in-between images. The dashed line shows the linear path of one feature during the course of the transformation. This example is indicative of the types of distortions that can arise with image morphing techniques.

ing a morph (image warp and cross-dissolve) between the prewarped images, and then postwarping each in-between image produced by the morph. The prewarping step is performed automatically, while the postwarping procedure may be interactively controlled by means of a small number of user-specified control points. Any of several image morphing techniques, for instance [15, 1, 8], may be used to compute the intermediate image interpolation. View morphing does not require knowledge of 3D shape, thereby allowing virtual manipulations of unknown objects or scenes given only as drawings or photographs.

In terms of its ability to achieve changes in viewpoint, view morphing is related to previous view-based techniques such as view synthesis [3, 7, 11, 12] and mosaics [10, 2, 14, 6]. However, this paper focuses on creating natural *transitions* between images rather than on synthesizing arbitrary views of an object or scene. This distinction has a number of important consequences. First, in computing the transition between two perspective views, we are free to choose a natural camera path. By choosing this path along the line connecting the two optical centers, we show that the formulation and implementation is greatly simplified. Second, our approach is general in that it can be used to compute transitions between any two images, thereby encompassing both rigid and nonrigid transformations. In contrast, previous view-based techniques have focused on rigid scenes. Finally, view morphing takes advantage of existing image morphing techniques, already in widespread use, for part of the computation. Existing image morphing tools may be easily extended to produce view morphs by adding the image prewarping and postwarping steps described in this paper.

The remainder of this paper is structured as follows: In Section 2 we review image morphing and argue that existing techniques may produce unnatural results when morphing between images of the same or similar shapes. Section 3 describes how to convert image morphing techniques into view morphing techniques by adding prewarping and postwarping steps. Section 4 extends the method to enable interpolations between views of arbitrary projective transformations of the same 3D object. In addition, interactive techniques for controlling the image transformations are introduced. We conclude with some examples in Section 5.

2 IMAGE MORPHING

Image morphing, or *metamorphosis*, is a popular class of techniques for producing transitions between images. There are a variety of morphing methods in the literature, all based on interpolating the positions and colors of pixels in two images. At present, there appears to be no universal criterion for evaluating the quality or realism of a morph, let alone of a morphing method. A natural question to ask, however, is does the method preserve 3D shape. That is, does a morph between two different views of an object produce new views of the same object? Our investigation indicates that unless special care is taken, morphing between images of similar 3D shapes often results in shapes that are mathematically quite different, leading to surprisingly complex and unnatural image transitions. These observations motivate *view morphing*, introduced in the next section, which preserves 3D shape under interpolation.

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. For example, an image point $(x, y) = \mathbf{p} \in \mathcal{I}$ is the projection of a scene point $(X, Y, Z) = \mathbf{P} \in \mathcal{S}$.

A morph is determined from two images $\mathcal{I}_0$ and $\mathcal{I}_1$ and maps $C_0 : \mathcal{I}_0 \Rightarrow \mathcal{I}_1$ and $C_1 : \mathcal{I}_1 \Rightarrow \mathcal{I}_0$ 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_0 and C_1 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 [15, 1, 8]. A warp function for each image is computed from the correspondence maps, usually based on linear interpolation:

$$W_0(\mathbf{p}_0, s) = (1-s)\mathbf{p}_0 + sC_0(\mathbf{p}_0) \quad (1)$$

$$W_1(\mathbf{p}_1, s) = (1-s)C_1(\mathbf{p}_1) + s\mathbf{p}_1 \quad (2)$$

W_0 and W_1 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 et al. [1] suggested two different methods of interpolating line features, using linear interpolation of endpoints, per Eqs. (1) and (2), or of position and angle. In this paper, the term *image morphing* refers specifically to methods that use linear interpolation to compute feature positions in in-between images, including [15, 1, 8].

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 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 since 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. 2 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 VIEW MORPHING

In the previous section we argued that unless special care is taken, image interpolations do not convey 3D rigid shape transformations. We say that an image transformation is *shape-preserving* if from two images of a particular object, it produces a new image representing a view of the same object. In this section we describe an interpolation-based image morphing procedure that is shape-preserving. Morphs generated by this technique create the illusion that the object moves rigidly (rotating and translating in 3D) between its positions in the two images.

Computing the morph requires the following: (1) two images $\mathcal{I}_0$ and $\mathcal{I}_1$, representing views of the same 3D object or scene, (2) their respective projection matrices $\mathbf{\Pi}_0$ and $\mathbf{\Pi}_1$, and (3) a correspondence between pixels in the two images. Note that no a priori knowledge of 3D shape information is needed. The requirement that projection matrices be known differentiates this technique from previous morphing methods. However, there exist a variety of techniques for obtaining the projection matrices from the images themselves and knowledge of either the internal camera parameters or the 3D positions of a small number of image points. For an overview of both types of techniques, consult [4]. In Section 4 we introduce a variant that does not require knowledge of the projection matrices and also allows interpolations between views of *different* 3D objects or scenes.

The pixel correspondences are derived by a combination of user-interaction and automatic interpolation provided by existing morphing techniques. When the correspondence is correct, the methods described in this section guarantee shape-preserving morphs. In practice, we have found that an approximate correspondence is often sufficient to produce transitions that are visually convincing. Major errors in correspondence may result in visible artifacts such as "ghosting" and shape distortions. Some examples of these effects are shown in Section 5. Other errors may occur as a result of changes in visibility. In order to completely infer the appearance of a surface from a new viewpoint, that surface must be visible in both $\mathcal{I}_0$ and $\mathcal{I}_1$. Changes in visibility may result in *folds* or *holes*, as discussed in Section 3.4.

Following convention, we represent image and scene quantities using homogeneous coordinates: a scene point with Euclidean coordinates (X, Y, Z) is expressed by the column vector $\mathbf{P} = [X\ Y\ Z\ 1]^T$ and a Euclidean image point (x, y) by $\mathbf{p} = [x\ y\ 1]^T$. We reserve the notation $\mathbf{P}$ and $\mathbf{p}$ for points

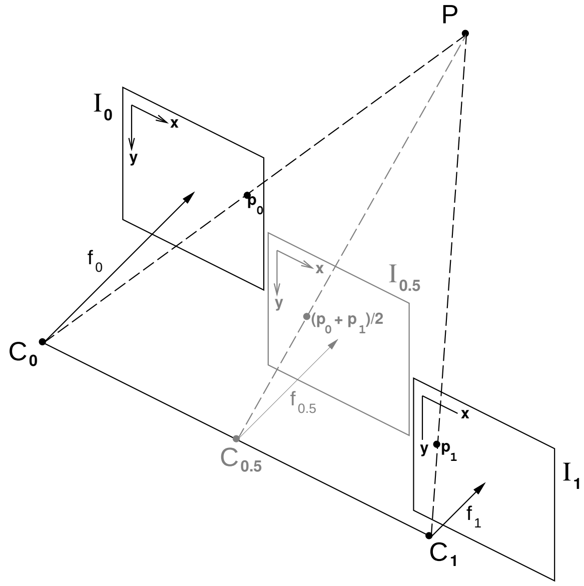

Figure 3: Morphing Parallel Views. Linear interpolation of corresponding pixels in parallel views with image planes $\mathcal{I}_0$ and $\mathcal{I}_1$ creates image $\mathcal{I}_{0.5}$, representing another parallel view of the same scene.

expressed in Euclidean coordinates, i.e., 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 3×4 homogeneous projection matrix of the form $\mathbf{\Pi} = [\mathbf{H} \mid -\mathbf{HC}]$. The vector $\mathbf{C}$ gives the Euclidean position of the camera's optical center and the 3×3 matrix $\mathbf{H}$ specifies the position and orientation of its image plane with respect to the world coordinate system. The perspective projection equation is

$$\tilde{\mathbf{p}} = \mathbf{\Pi}\mathbf{P} \tag{3}$$

The term *view* will henceforth refer to the tuple $\langle \mathcal{I}, \mathbf{\Pi} \rangle$ comprised of an image and its associated projection matrix.

3.1 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 $\mathcal{I}_1$, as shown in Fig. 3. Alternatively, we could produce the same two images by moving the camera instead of the object. Chen and Williams [3] 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:

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

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

We refer to cameras or views with projection matrices in this form as *parallel cameras* or *parallel views*, respectively.

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

$$\begin{aligned}(1-s)\mathbf{p}_0 + s\mathbf{p}_1 &= (1-s)\frac{1}{Z}\mathbf{\Pi}_0\mathbf{P} + s\frac{1}{Z}\mathbf{\Pi}_1\mathbf{P} \\ &= \frac{1}{Z}\mathbf{\Pi}_s\mathbf{P} \qquad (4)\end{aligned}$$

where

$$\mathbf{\Pi}_s = (1-s)\mathbf{\Pi}_0 + s\mathbf{\Pi}_1 \qquad (5)$$

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 center $\mathbf{C}_s$ and focal length f_s given by:

$$\begin{aligned}\mathbf{C}_s &= (sC_X, sC_Y, 0) \qquad (6)\\ f_s &= (1-s)f_0 + sf_1 \qquad (7)\end{aligned}$$

Consequently, interpolating images produced from parallel cameras produces the illusion of simultaneously moving the 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.

In fact, the above derivation 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.

3.2 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.

3.2.1 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 3×3 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 3×3 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

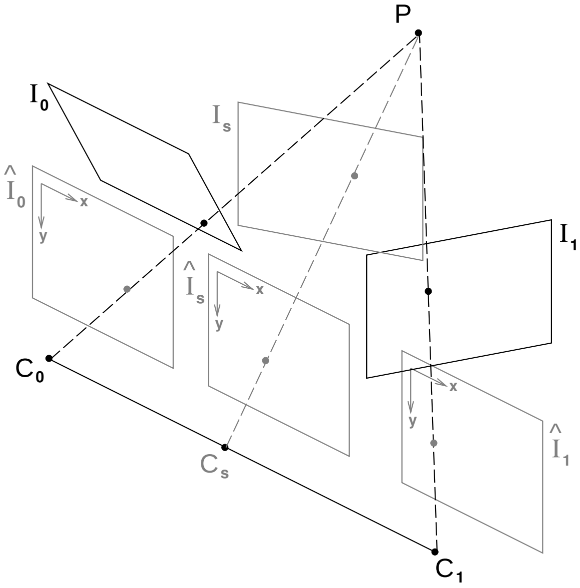

Figure 4: View Morphing in Three Steps. (1) Original images $\mathcal{I}_0$ and $\mathcal{I}_1$ are prewarped to form parallel views $\hat{\mathcal{I}}_0$ and $\hat{\mathcal{I}}_1$. (2) $\hat{\mathcal{I}}_s$ is produced by morphing (interpolating) the prewarped images. (3) $\hat{\mathcal{I}}_s$ is postwarped to form $\mathcal{I}_s$.

algorithm [15]. 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 [15]. Our use of reprojection is most closely related to the techniques used for rectifying stereo views to simplify 3D shape reconstruction [4]. Image mosaic techniques [10, 2, 14, 6] 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.

3.2.2 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 3.1. 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 reduces the distortion incurred by image reprojection. 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. 4. Given a projection matrix $\mathbf{\Pi}_s = [\mathbf{H}_s \mid -\mathbf{H}_s\mathbf{C}_s]$, with $\mathbf{C}_s$ fixed by Eq. (6), 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. (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 3×3 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 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.

The prewarping and postwarping operations, combined with the intermediate morph, require multiple image resampling operations that may contribute to a noticeable blurring in the in-between images. Resampling effects can be reduced by supersampling the input images [15] 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 [1], 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.

3.3 Singular View Configurations

Certain configurations of views cannot be made parallel through reprojection operations. For parallel cameras, (Fig. 5, 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[1]. Fig. 5 (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 the Appendix). Singular views are not a problem when the prewarp, morph, and postwarp are composed into a single aggregate warp, since prewarped images are never explicitly constructed. With aggregate warps, view morphing may be applied to arbitrary pairs of views, including singular views.

3.4 Changes in Visibility

So far, we have described how to correct for distortions in image morphs by manipulating the projection equations. Eq. (3), however, does not model the effects that changes

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

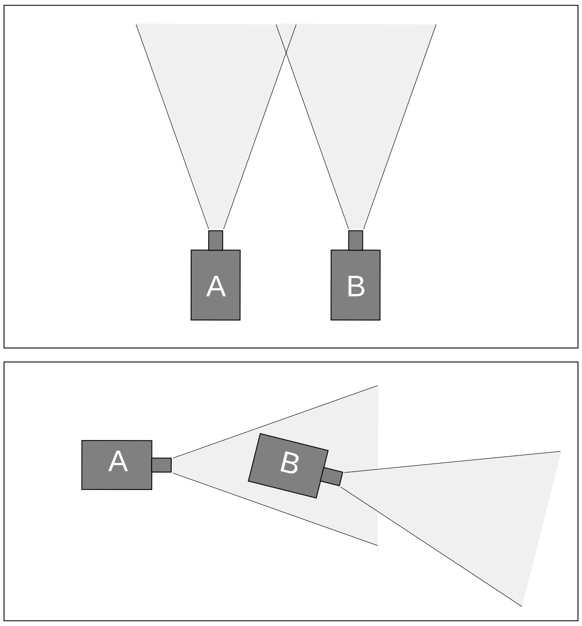

Figure 5: Singular Views. In the parallel configuration (top), each camera's optical center is out of the field of view of the other. A singular configuration (bottom) arises when the optical center of camera B is in the field of view of camera A. Because prewarping does not change the field of view, singular views cannot be reprojected to form parallel views.

in *visibility* have on image content. 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 [3], 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. 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, 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 [10].

Unlike folds, holes cannot always be eliminated using image information alone. Chen and Williams [3] 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.

3.4.1 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. (6), 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 θ 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 [13]. In either case, camera parameters such as focal length and aspect ratio should be interpolated separately.

4 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. 2.

As described in Section 3.1, a 2D projective transformation may be expressed as a 3×3 homogeneous matrix transformation. Similarly, a 3D projective transformation is given by a 4×4 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 3×4 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 3×4 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 3.2.2 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.

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 3.2.2, 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. Consequently $\mathcal{I}_0$ and $\mathcal{I}_1$ may be made parallel by finding any pair of 2D projective transformations $\mathbf{H}_0$ and $\mathbf{H}_1$ 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 the Appendix.

4.1 Controlling the Morph

To fully determine a view morph, $\mathbf{H}_s$ must be provided for each in-between image. Rather than specifying the 3×3 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 region[2]. Fig. 6 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 positions of the control points in $\mathcal{I}_s$ and $\hat{\mathcal{I}}_s$ specify a linear system of equations whose solution yields $\mathbf{H}_s$ [15]. The four curves traced out by the control points may also be manually edited for finer control of the interpolation parameters.

The use of image control points bears resemblance to the view synthesis work of Laveau and Faugeras [7], 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.

4.2 View Morphing Without Prewarping

Prewarping is less effective for morphs between different objects not closely related by a 3D projective transform. With objects that are considerably different, it is advisable to leave out the prewarp entirely, since its automatic computation becomes less stable [9]. The postwarp step should not be omitted, however, since it can be used to reduce image plane distortions for more natural morphs. For instance, a large change in orientation results in a noticeable 2D image contraction, as seen in Fig. 10.

[2] Care should be taken to ensure that no three of the control points are colinear in any image.

Prewarping is not strictly necessary for images that are approximately orthographic, as noted in Section 3.1. Images taken with a telephoto lens often fall into this category, as do images of objects whose variation in depth is small relative to their distance from the camera. In either case, the images may be morphed directly, yielding new orthographic views. However, the prewarping step does influence the camera motion which, in the orthographic case, cannot be controlled solely by postwarping. The camera transformation determined by Eq. (5) may introduce unnatural skews and twists of the image plane due to the fact that linear matrix interpolation does not preserve row orthogonality. Prewarping the images ensures that the view plane undergoes a single axis rotation. More details on the orthographic case are given in [12].

5 RESULTS

Fig. 6 illustrates the view morphing procedure applied to two images of a bus. We manually selected a set of about 50 corresponding line features in the two images. These features were used to automatically prewarp the images to achieve parallelism using the method described in the Appendix. Inspection of the prewarped images confirms that corresponding features do in fact occupy the same scanlines. An implementation of the Beier-Neely field-morphing algorithm [1] 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. 6 were computed automatically by roughly calibrating the two images based on their known focal lengths and interpolating the changes in orientation [4]. Different images obtained by other settings of the control points are shown in Fig. 8. As these images indicate, a broad range of 3D projective effects may be achieved through the postwarping procedure. For instance, the rectangular shape of the bus can be skewed in different directions and tapered to depict different 3D shapes.

Fig. 7 shows some results on 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.

When an object has bilateral symmetry, view morphs can be computed from a single image. Fig. 9 depicts a view morph between an image of Leonardo da Vinci's *Mona Lisa* and its mirror reflection. Although the two sides of the face and torso are not perfectly symmetric, the morph conveys a convincing facial rotation.

Fig. 10 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. 2 are also present. Image morphing techniques such as [1] 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. To be sure, view morphs may also produce holes, but only as a result of a change in visibility.

6 CONCLUSIONS

Achieving realistic image transitions is possible but often difficult with existing image morphing techniques due to the lack of available 3D information. In this paper, we demonstrated how to accurately convey a range of 3D transformations based on a simple yet powerful extension to the image morphing paradigm called *view morphing.* In addition to changes in viewpoint, view morphing accommodates changes in projective shape. By integrating these capabilities with those already afforded by existing image morphing methods, view morphing enables transitions between images of different objects that give a strong sense of metamorphosis in 3D. Because no knowledge of 3D shape is required, the technique may be applied to photographs and drawings, as well as to artificially rendered scenes. Two different methods for controlling the image transition were described, using either automatic interpolation of camera parameters or interactive user-manipulation of image control points, based on whether or not the camera viewpoints are known.

Because view morphing relies exclusively on image information, it is sensitive to changes in visibility. In our experiments, the best morphs resulted when visibility was nearly constant, i.e., most surfaces were visible in both images. The visible effects of occlusions may often be minimized by experimenting with different feature correspondences. Additional user input could be used to reduce ghosting effects by specifying the paths of image regions visible in only one of the original images. A topic of future work will be to investigate ways of extending view morphing to handle extreme changes in visibility, enabling 180 or 360 degree rotations in depth.

ACKNOWLEDGEMENTS

The authors would like to thank Paul Heckbert for providing the image morphing code used to interpolate prewarped images in this paper. The original helicopter images in Fig. 10 were rendered using Rayshade with a Z-buffer output extension written by Mark Maimone. The support of the National Science Foundation under Grant Nos. IRI–9220782 and CDA–9222948 is gratefully acknowledged.

REFERENCES

[1] Beier, T., and Neely, S. Feature-based image metamorphosis. Proc. SIGGRAPH 92. In *Computer Graphics* (1992), pp. 35–42.

[2] Chen, S. E. Quicktime VR — An image-based approach to virtual environment navigation. Proc. SIGGRAPH 95. In *Computer Graphics* (1995), pp. 29–38.

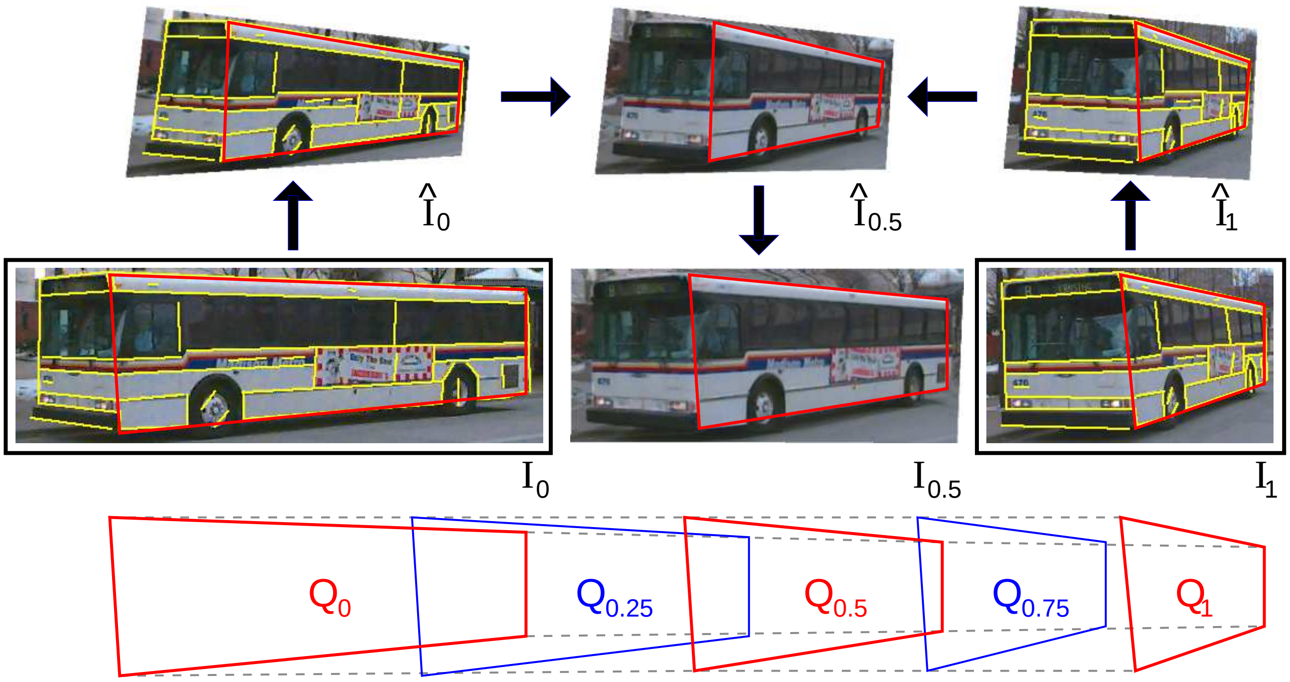

Figure 6: View Morphing Procedure: A set of features (yellow lines) is selected in original images $\mathcal{I}_0$ and $\mathcal{I}_1$. Using these features, the images are automatically prewarped to produce $\hat{\mathcal{I}}_0$ and $\hat{\mathcal{I}}_1$. The prewarped images are morphed to create a 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).

$\mathcal{I}_0$ $\mathcal{I}_{0.25}$ $\mathcal{I}_{0.5}$ $\mathcal{I}_{0.75}$ $\mathcal{I}_1$

Figure 7: 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.

Figure 8: Postwarping deformations obtained by different settings of the control quadrilateral.

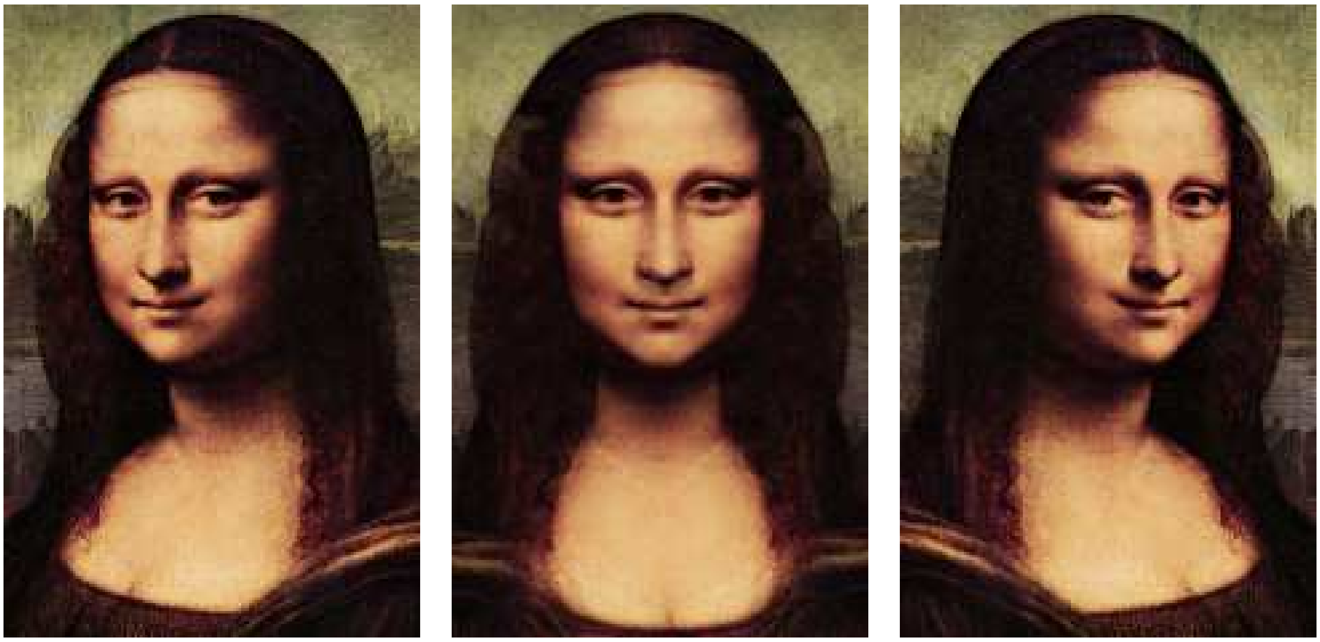

Figure 9: Mona Lisa View Morph. Morphed view (center) is halfway between original image (left) and it's reflection (right).

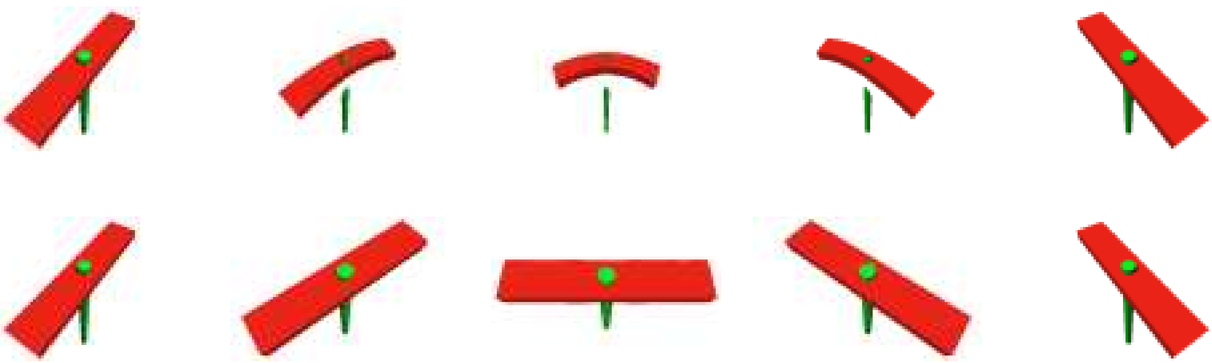

Figure 10: Image Morphing Versus View Morphing. Top: image morph between two views of a helicopter toy causes the in-between images to contract and bend. Bottom: view morph between the same two views results in a physically consistent morph. In this example the image morph also results in an extraneous hole between the blade and the stick. Holes can appear in view morphs as well.

[3] Chen, S. E., and Williams, L. View interpolation for image synthesis. Proc. SIGGRAPH 93. In *Computer Graphics* (1993), pp. 279–288.

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

[5] Hartley, R. I. In defence of the 8-point algorithm. In *Proc. Fifth Intl. Conference on Computer Vision* (1995), pp. 1064–1070.

[6] Kumar, R., Anandan, P., Irani, M., Bergen, J., and Hanna, K. Representation of scenes from collections of images. In *Proc. IEEE Workshop on Representations of Visual Scenes* (1995), pp. 10–17.

[7] Laveau, S., and Faugeras, O. 3-D scene representation as a collection of images. In *Proc. International Conference on Pattern Recognition* (1994), pp. 689–691.

[8] Lee, S.-Y., Chwa, K.-Y., Shin, S. Y., and Wolberg, G. Image metamorphosis using snakes and free-form deformations. Proc. SIGGRAPH 92. In *Computer Graphics* (1992), pp. 439–448.

[9] Luong, Q.-T., and Faugeras, O. The fundamental matrix: Theory, algorithms, and stability analysis. *Intl. Journal of Computer Vision 17*, 1 (1996), 43–75.

[10] McMillan, L., and Bishop, G. Plenoptic modeling. Proc. SIGGRAPH 95. In *Computer Graphics* (1995), pp. 39–46.

[11] Poggio, T., and Beymer, D. Learning networks for face analysis and synthesis. In *Proc. Intl. Workshop on Automatic Face- and Gesture-Recognition* (Zurich, 1995), pp. 160–165.

[12] Seitz, S. M., and Dyer, C. R. Physically-valid view synthesis by image interpolation. In *Proc. IEEE Workshop on Representations of Visual Scenes* (1995), pp. 18–25.

[13] Shoemake, K. Animating rotation with quaternion curves. Proc. SIGGRAPH 85. In *Computer Graphics* (1985), pp. 245–254.

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

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

APPENDIX

This appendix describes how to automatically compute the image prewarping transforms $\mathbf{H}_0$ and $\mathbf{H}_1$ from the images themselves. We assume that the 2D positions of 8 or more corresponding points are given in each image. The fundamental matrix of two views is defined to be the 3×3, rank-two matrix $\mathbf{F}$ such that for every pair of corresponding image points $\mathbf{p}_0 \in \mathcal{I}_0$ and $\mathbf{p}_1 \in \mathcal{I}_1$,

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

$\mathbf{F}$ is defined up to a scale factor and can be computed from 8 or more such points using linear [5] or non-linear [9] methods.

A sufficient condition for two views to be parallel is that their fundamental matrix have the form:

$$\hat{\mathbf{F}} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \tag{8}$$

Consequently, any two images with fundamental matrix $\mathbf{F}$ may be prewarped (i.e., made parallel) by choosing any two projective transforms $\mathbf{H}_0$ and $\mathbf{H}_1$ such that $(\mathbf{H}_1^{-1})^T \mathbf{F} \mathbf{H}_0^{-1} = \hat{\mathbf{F}}$. Here we describe one method that applies a rotation in depth to make the images planes parallel, followed by an affine transformation to align corresponding scanlines. The procedure is determined by choosing an (arbitrary) axis of rotation $\mathbf{d}_0 = [d_0^x \; d_0^y \; 0]^T \in \mathcal{I}_0$. Given $[x \; y \; z]^T = \mathbf{F}\mathbf{d}_0$, the corresponding axis in $\mathcal{I}_1$ is determined according to $\mathbf{d}_1 = [-y \; x \; 0]^T$. To compute the angles of depth rotation we need the *epipoles*, also known as *vanishing points*, $\mathbf{e}_0 \in \mathcal{I}_0$ and $\mathbf{e}_1 \in \mathcal{I}_1$. $\mathbf{e}_0 = [e_0^x \; e_0^y \; e_0^z]^T$ and $\mathbf{e}_1 = [e_1^x \; e_1^y \; e_1^z]^T$ are the unit eigenvectors of $\mathbf{F}$ and $\mathbf{F}^T$ respectively, corresponding to eigenvalues of 0. A view's epipole represents the projection of the optical center of the other view. The following procedure will work provided the views are not *singular*, i.e., the epipoles are outside the image borders and therefore not within the field of view. The angles of rotation in depth about $\mathbf{d}_i$ are given by

$$\theta_i = -\frac{\pi}{2} - tan^{-1}\left(\frac{d_i^y e_i^x - d_i^x e_i^y}{e_i^z}\right)$$

We denote as $\mathbf{R}_{\theta_i}^{\mathbf{d}_i}$ the 3×3 matrix corresponding to a rotation of angle θ_i about axis $\mathbf{d}_i$. 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. 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 [15].

The next step is to rotate the images so that epipolar lines are horizontal. The new epipoles are $[\tilde{e}_i^x \; \tilde{e}_i^y \; 0]^T = \mathbf{R}_{\theta_i}^{\mathbf{d}_i} \mathbf{e}_i$. The angles of rotation ϕ_0 and ϕ_1 are given by $\phi_i = -tan^{-1}(\tilde{e}_i^y / \tilde{e}_i^x)$. After applying these image plane rotations, the fundamental matrix has the form

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

The 3×3 matrix $\mathbf{R}_\theta$ denotes an image plane (z axis) rotation of angle θ. Finally, to get $\mathbf{F}$ into the form of Eq. (8), the second image is translated and vertically scaled by the matrix

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

In summary, the prewarping transforms $\mathbf{H}_0$ and $\mathbf{H}_1$ are

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

The entire procedure is determined by selecting $\mathbf{d}_0$. A suitable choice is to select $\mathbf{d}_0$ orthogonal to $\mathbf{e}_0$, i.e., $\mathbf{d}_0 = [-e_0^y \; e_0^x \; 0]^T$.