Chapter 1

DYNAMIC VIEW INTERPOLATION WITHOUT AFFINE RECONSTRUCTION

Russell A. Manning
Charles R. Dyer

Abstract This chapter presents techniques for view interpolation between two reference views of a dynamic scene captured at different times. The interpolations produced portray one possible physically-valid version of what transpired in the scene during the time between when the two reference views were taken. We show how straight-line object motion, relative to a camera-centered coordinate system, can be achieved, and how the appearance of straight-line object motion relative to the background can be created. The special case of affine cameras is also discussed. The methods presented work with widely-separated, uncalibrated cameras and sparse point correspondences. The approach does not involve finding the camera-to-camera transformation and thus does not implicitly perform affine reconstruction of the scene. For circumstances in which the camera-to-camera transformation can be found, we introduce a vector-space of possible synthetic views that follows naturally from the given reference views. It is assumed that the motion of each object in the original scene consists of a series of rigid translations.

1. INTRODUCTION

View interpolation [5] involves creating a sequence of virtual views of a scene that, taken together, represent a continuous and physically-correct transition between two reference views. A number of techniques for view interpolation have been developed to date, but all have been restricted to static scenes. Dynamic scenes change over time and, consequently, these changes will be evident in two reference views that are captured at different times. Therefore, view interpolation for dynamic scenes must

*To appear in *Confluence of Computer Vision and Computer Graphics*, A. Leonardis, F. Solina, and R. Bajcsy, eds., Kluwer, Dordrecht, The Netherlands, 2000

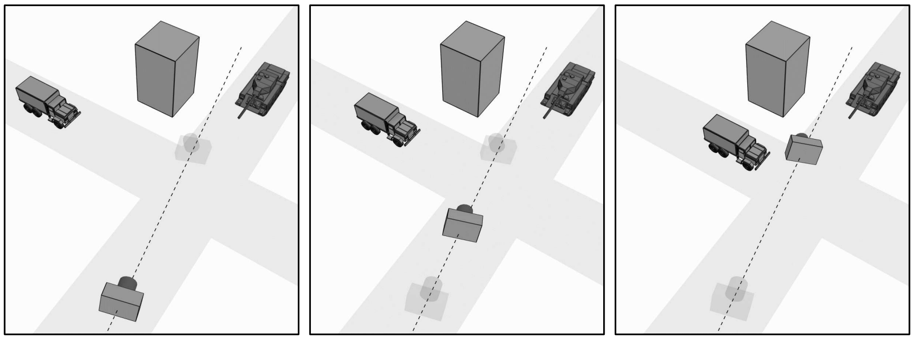

Figure 1.1 View interpolation of a dynamic scene involves creating the view for the virtual camera in the middle frame using only the two reference views from the cameras in the left and right frames.

portray a continuous change in viewpoint *and* a continuous change in the scene itself in order to transition smoothly between the reference views.

Algorithms that can interpolate and extrapolate the motion of dynamic scenes from a set of reference views have a wide range of potential applications. Using such an algorithm, video compression and video summarization could be accomplished by the selection of a small number of key frames bracketing primitive motion events. Dynamic augmented-reality sequences could be created that place moving synthetic objects into real scenes. Video sequences could be generated from legacy photographs of dynamic scenes. Missing gaps in video sequences, arising from camera slews, could be filled in (as illustrated in Fig. 1.1). Occluding objects, both stationary and moving, could be removed from video sequences. Finally, the ability to perform temporal extrapolation of a dynamic environment could lead to a wide variety of important applications in visual servoing, navigation, and robot motion planning (where visual simulations of objects in motion could be used to anticipate the effects of actions). It is interesting to note that the human visual system is capable of generating the impression of smooth, continuous apparent motion when shown alternating images at appropriate intervals [21], further motivation for this area of research.

Our approach to view interpolation for dynamic scenes is called *dynamic view morphing* [17]. The method is based on our earlier work for static scenes called "view morphing" [22] and it retains the advantages of that earlier work. In particular, only two reference views of the scene are required, each reference camera can be different and uncalibrated, the views can be widely separated, and only a sparse set of corresponding points between the views is necessary. If more information about the reference views is available, this information can be used for added control over the output and for increased realism.

Dynamic view morphing follows a pure image-based rendering (IBR) methodology wherein the goal is to compute new views directly from the input images with minimal recovery of three-dimensional information about the cameras or scene. Such direct approaches offer several advantages over image-based methods that involve explicit scene reconstruction. For instance, pure IBR methods avoid potentially noise-producing intermediate steps such as the estimation of camera calibration. They can be computationally efficient, operating independently of scene complexity and number of input views. They can produce photorealistic results in cases where explicit reconstruction methods are not reliable (e.g., with small scene objects or noisy images). Their output can be gracefully degraded when computational efficiency is more important than exact photorealism (e.g., when creating an out-of-focus background for a movie special-effect shot). Finally, they do not amplify errors in camera calibration and conjugate point correspondences as many explicit reconstruction algorithms, by repeated reconstruction and reprojection of scene points, are prone to doing.

Methods exist for creating view interpolations of static scenes without explicit scene reconstruction [1, 5, 22, 25] or with (at most) explicit projective reconstruction [9]. Many other methods exist for performing explicit affine or metric reconstruction of static scenes (e.g., [24, 18, 23, 16]). Once explicit scene models have been recovered, view interpolations can be created via the standard graphics pipeline. Of particular recent interest in the area of scene reconstruction are a variety of "self calibration" techniques (e.g., [10, 11, 13, 19, 20]) which can perform affine or metric scene reconstruction without relying on scene knowledge to calibrate the cameras. Unfortunately, all of the techniques listed so far were developed specifically for static scenes and cannot be directly applied to the kinds of dynamic scenes considered in this chapter.

Some related work has been performed specifically for dynamic scenes. Avidan and Sashua [2] developed a method for explicitly recovering the geometry of dynamic scenes in which the objects move along straight-line trajectories. However, their algorithm does not apply to the problem discussed in this chapter because it assumes that five or more views are available and that the camera matrix for each view is known or can be recovered. Clarke and Zisserman [6] developed a related method for the problem of collision detection; they also required the computation of the camera matrix for each object and view. There are several mosaicing techniques for dynamic scenes [7, 14], but mosaicing involves compositing multiple small-field views to create a single large-field view, whereas view interpolation synthesizes new views from vantage points not in the input set. Non-reconstructive image-based rendering methods have been applied to animating facial expressions (e.g., [4, 8, 26]) and body motions [15], though these methods were not concerned with creating physically-correct new views.

The techniques of dynamic view morphing presented in this chapter apply only to scenes that satisfy the following assumption: For each object in the scene, all of the changes that the object undergoes during the missing time interval, when taken together, are equivalent to a single, rigid translation of the object.

The term *object* has a specific meaning in this chapter, defined by the condition given above: An object is a set of points in a scene for which there exists a fixed vector $\mathbf{u} \in \Re^3$ such that each point's total motion during the missing time interval is equal to $\mathbf{u}$.

When the *camera-to-camera transformation* (see Section 3.) between the reference views can be determined, the synthetic view interpolations produced by dynamic view morphing will portray *linear motion*. This means the synthetic interpolation sequence will portray all objects in the scene as undergoing rigid, constant-velocity, straight-line motion as viewed from a virtual camera moving along a straight-line, constant-velocity trajectory.

Unfortunately, calculating the camera-to-camera transformation is at least as difficult as affine reconstruction of objects in the scene. Consequently, the focus of this chapter is on techniques that do *not* require knowledge of the camera-to-camera transformation. We demonstrate how a variety of conditions can be imposed on the interpolations without guaranteeing linear motion. In the special case of affine cameras, we discuss how the camera-to-camera transformation cannot be found (except in special cases) and what this means for the resulting view interpolations.

2. DYNAMIC VIEW MORPHING

Dynamic view morphing refers to a variety of techniques whose applicability depends on what information can be extracted from the reference views. All of the techniques share the following structure:

The reference views are segmented into layers, with each layer representing an object or part of an object.

Each object layer is interpolated separately, and the resulting intermediate views of each object are combined to produce the final, complete intermediate view of the whole scene.

The second step above (interpolation of individual object layers) proceeds as follows:

A prewarp transformation is applied to make the image planes of both reference views parallel to each other and to the object's motion.

Conjugate points are linearly interpolated to produce a physically-accurate intermediate view (of the conjugate points only).

The intermediate view is completed via a morphing algorithm (e.g., Beier-Neely [3]) using the interpolated conjugate points as guides.

3. NOTATION AND PRELIMINARY CONCEPTS

This and the following section provide the mathematical background for dynamic view morphing. In particular it is shown that, with proper prewarping of the reference views, linear interpolation of conjugate points produces a correct new view of the scene.

Assume two reference views are captured at times $t = 0$ and $t = 1$ using pinhole cameras, which are denoted *camera* A and *camera* B, respectively. A *fixed-camera formulation* is used, meaning the two reference cameras are treated as if they were at the same location and the world is moving around them; this is accomplished by subtracting the displacement between the two cameras from the motion vectors of all objects in the scene. When the scene is static, there will be a single moving object, called the background object, consisting of all points in the scene. Under the fixed-camera assumption, the camera matrices are just 3×3 and each camera is equivalent to a basis for $\Re^3$. Note that no assumption is made about the cameras other than they share the same optical center; the camera matrices can be completely different.

Let U denote the "universal" or "world" coordinate frame, and let the notation $\mathbf{H}^{\infty}_{UA}$ mean the transformation between basis U and basis A. Hence $\mathbf{H}^{\infty}_{UA}$ is the camera matrix for A. Note that boldface capital letters will always represent 3×3 matrices; in particular, $\mathbf{I}$ will be the identity matrix.

Of particular interest is the matrix $\mathbf{H}^{\infty}_{AB}$. This matrix represents a homography between the planes at infinity of the two cameras; that is, it transforms a direction as represented in one camera into the same direction represented in the other camera. The notation $\mathbf{H}^{\infty}$ signifies how the basis transformation equals the infinity homography. Under the fixed-camera formulation $\mathbf{H}^{\infty}_{AB}$ transforms the view from camera A into the view from camera B. For this reason, we refer to it as the *camera-to-camera transformation.*

A position or a direction in space exists independently of what basis is used to measure it; we will use a subscript letter when needed to denote a particular basis. For instance, if $\mathbf{e}$ is the direction between two camera's optical centers (ignoring the fixed-camera formulation momentarily), then $\mathbf{e}_A$ is $\mathbf{e}$ measured in basis A. The quantity $\mathbf{e}$ is called the *epipole*. The fundamental matrix $\mathbf{F}$ for two cameras A and B that are at different locations has the following representation [12]:

$$\mathbf{F} = [\mathbf{e}_B]_\times \mathbf{H}^\infty_{AB} \tag{1.1}$$

where $[\cdot]_\times$ denotes the cross product matrix. When the two cameras share the same optical center, the fundamental matrix is 0 and has no meaning. However, for each moving object Ω in the scene, we can define a new kind of fundamental matrix. If, after making the fixed-camera assumption, Ω is moving in direction $\mathbf{u}$, then the fundamental matrix *for the object* is:

$$\mathbf{F}_\Omega = [\mathbf{u}_B]_\times \mathbf{H}^\infty_{AB} \tag{1.2}$$

The epipoles of $\mathbf{F}_\Omega$ are the vanishing points of Ω as viewed from the two reference cameras, and the epipolar lines trace out trajectories for points on Ω.

4. VIEW INTERPOLATION FOR A SINGLE MOVING OBJECT

Assume the two reference cameras share the same optical center and are viewing a point ω that is part of an object Ω whose translation vector is $\mathbf{u}$. Let $\mathbf{q}$ and $\mathbf{q}+\mathbf{u}$ denote the position of ω at times $t=0$ and $t=1$, respectively (Fig. 1.2).

Assume for this section that the image planes of the cameras are parallel to each other and to $\mathbf{u}$. The first half of this condition means that the third row of $\mathbf{H}^\infty_{UA}$ equals the third row of $\mathbf{H}^\infty_{UB}$ scaled by some constant λ. The second half means that $(\mathbf{H}^\infty_{UA}\mathbf{u}_U)_z = (\mathbf{H}^\infty_{UB}\mathbf{u}_U)_z = 0$, where $(\cdot)_z$ denotes the z-coordinate of a vector. Note that the condition of parallelism can be met retroactively by using standard rectification methods [22].

Setting $\xi = (\mathbf{H}^\infty_{UA}\mathbf{q}_U)_z = \lambda(\mathbf{H}^\infty_{UB}(\mathbf{q}+\mathbf{u})_U)_z$, the linear interpolation of the projection of ω into both cameras is

$$(1-s)\frac{1}{\xi}\mathbf{H}^\infty_{UA}\mathbf{q}_U + s\frac{\lambda}{\xi}\mathbf{H}^\infty_{UB}(\mathbf{q}+\mathbf{u})_U \tag{1.3}$$

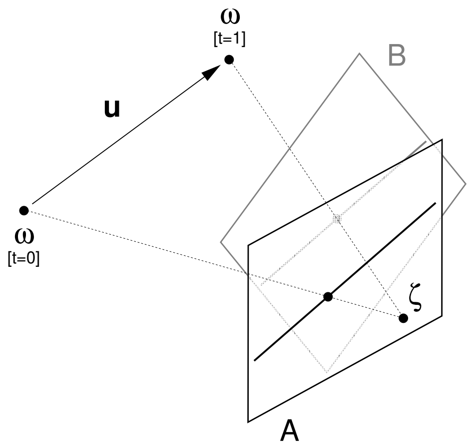

Figure 1.2 Cameras A and B share the same optical center ζ and are viewing a point on an object that translates by $\mathbf{u}$. The image planes of the cameras are parallel to each other and to $\mathbf{u}$, and hence interpolation will produce a physically-correct view of the object. On each image plane a line parallel to $\mathbf{u}$ is shown.

Now define a virtual camera V by the matrix

$$\mathbf{H}_{UV}^{\infty} = (1-s)\mathbf{H}_{UA}^{\infty} + s\lambda\mathbf{H}_{UB}^{\infty} \tag{1.4}$$

Then the linear interpolation (Eq. 1.3) is equal to the projection of scene point $\mathbf{q}(s)$ onto the image plane of camera V, where

$$\begin{aligned} \mathbf{q}(s) &= \mathbf{q} + \mathbf{u}(s) && (1.5) \\ \mathbf{u}(s)_V &= s\lambda\mathbf{u}_B && (1.6) \end{aligned}$$

Notice that $\mathbf{u}(s)$ depends only on $\mathbf{u}$ and the camera matrices and not on the starting location $\mathbf{q}$. Thus linear interpolation of conjugate object points by a factor s creates a physically-valid view of the object. The object is seen as it would appear through camera V if it had translated by $\mathbf{u}(s)$ from its starting position.

Note that in Eq. 1.6, $\mathbf{u}(s)$ is represented in basis V. Since V changes with s it is difficult in general to characterize the trajectory in world coordinates. A key result in [17] states that, if $\mathbf{u}_A = k\mathbf{u}_B$, then

$$\mathbf{u}(s)_B = \frac{s\lambda}{(1-s)k + s\lambda}\mathbf{u}_B \tag{1.7}$$

Thus the virtual trajectory, given by $\mathbf{u}(s)_B$, is a straight-line in basis B; basis V no longer plays a role. If $k = \lambda$ then $\mathbf{u}(s)_B = s\mathbf{u}_B$ and the object moves at constant velocity. The results are in basis B, but multiplying by $\mathbf{H}_{BU}^{\infty}$ or $\mathbf{H}_{BA}^{\infty}$ indicates that the results also hold in world coordinates and camera A's coordinates. Keep in mind that the world coordinate system used in this context has its origin at the shared optical center of the reference cameras.

5. INTERPOLATION WITHOUT H^{∞}

When $\mathbf{H}_{AB}^{\infty}$ can be determined, the dynamic view interpolation will portray linear motion [17]. Unfortunately, finding $\mathbf{H}_{AB}^{\infty}$ can be a difficult task. In fact, once $\mathbf{H}_{AB}^{\infty}$ is known, each scene object can be reconstructed up to an affine transformation. The affine transformation is the same for each object up to an unknown scale factor that differs for each object. Hence, finding $\mathbf{H}_{AB}^{\infty}$ is at least as difficult as affine object reconstruction.

In this section we discuss the problem of creating dynamic view interpolations without first calculating $\mathbf{H}_{AB}^{\infty}$. All of the methods presented produce physically-valid view interpolations; some portray straight-line motion relative to a camera-centered coordinate frame. None are guaranteed to portray constant-velocity motion, however.

In addition, we show how the *appearance* of straight-line motion can be produced in certain situations by correct prewarping. Finally, we discuss the benefits and drawbacks of using the methods in conjunction with affine views.

5.1 SPECIAL CASE: PARALLEL MOTION

Assume a fixed-camera formulation and let $\mathbf{u}_i$ denote the 3D displacement between the position of object i at time $t = 0$ and its position at time $t = 1$. We will say the scene consists of *parallel motion* if all the $\mathbf{u}_i$ are parallel in scene space.

Dynamic view morphing algorithm for the parallel motion case: *Segment each view into layers corresponding to objects. Apply static view morphing to each layer and recomposite the results.*

The algorithm works because the fundamental matrix with respect to each object is the same, so the same prewarp works for each layer. The prewarp will make the direction of motion for each object parallel to the x-axis in both views; consequently, the objects will follow straight-line trajectories as measured in the camera frame. If we assume that the background object has no motion in world coordinates, then the virtual camera also moves parallel to the motion of all the objects and hence each object's motion is straight-line in world coordinates.

5.2 SPECIAL CASE: PLANAR PARALLEL MOTION

We now consider the case in which all the $\mathbf{u}_i$ are parallel to some fixed plane in space. Note that this does not mean all the objects are translating in the *same* plane. Also note that this case applies whenever there are two moving objects (i.e., the background object and one object moving in the scene).

Recall that in Section 4. the only requirement for the virtual view to be a physically-accurate portrayal of an object that translates by $\mathbf{u}$ is that the image planes of both reference views be parallel to $\mathbf{u}$ and to each other. In the planar parallel motion case, it is possible to prewarp the reference views so that their image planes are parallel to each other and to the displacements of all objects simultaneously.

Dynamic view morphing algorithm for the planar parallel motion case: *Segment each view into layers corresponding to objects. For each reference view, find a single prewarp that sends the z coordinate of the vanishing point of each object to* 0. *Using this prewarp, apply static view morphing to each layer and then composite the results.*

The algorithm given above only guarantees physical correctness, not straight-line or linear motion. The *appearance* of straight-line motion can be created by first making the conjugate motion vectors parallel during the prewarp step.

5.3 PRODUCING THE APPEARANCE OF STRAIGHT MOTION WITHOUT H^{∞}

We now give an example to illustrate the planar parallel motion algorithm and to make an important observation (Fig. 1.3). This example serves to illustrate the technique for producing straight-line trajectories with affine cameras as well. For the example, assume a car is driving down a straight road on level terrain and is photographed at two different times from an airplane that is not moving parallel to the car but is flying at constant altitude. The flat surface of the ground can be taken as the common parallel plane.

Assuming the fixed-camera formulation, let $\mathbf{u}_0$ denote the direction of motion of the background object and let $\mathbf{u}_1$ denote the direction the road is heading. That is, $\mathbf{u}_1$ is a vector parallel to the edges of the road. Note that $\mathbf{u}_1$ is not the direction of the car's motion under the fixed-camera formulation; the car's motion is the original translation of the car traveling down the road minus the translation of the camera.

Since vectors $\mathbf{u}_0$ and $\mathbf{u}_1$ are both parallel to the ground, they can be used to transform the two reference views to be parallel to the common plane. After this transformation, the views can be further transformed so that $\mathbf{u}_0$ points in the direction of the y-axis in both views and $\mathbf{u}_1$ points in the direction of the x-axis.

The car's translation at this point is the sum of two perpendicular vectors: one is the translation of the background, which is entirely along

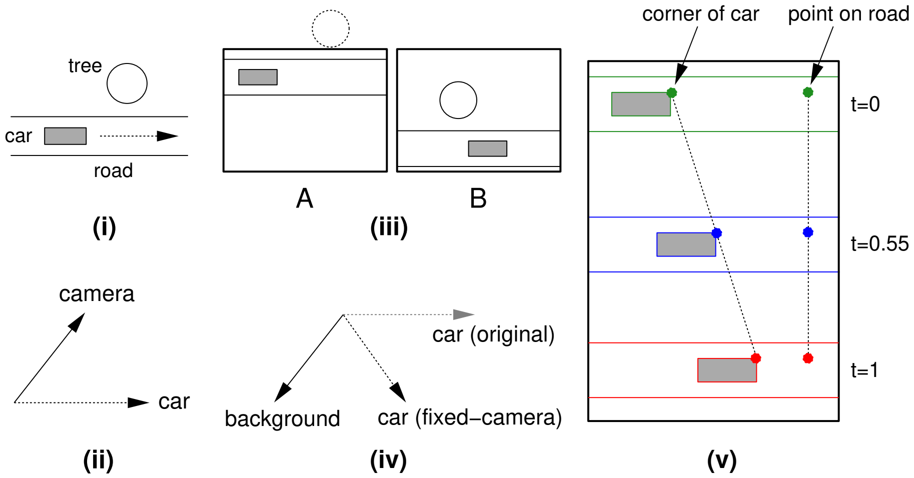

Figure 1.3 Illustration of how a car object can appear to stay traveling down a road during an interpolation sequence even when $\mathbf{H}_{AB}^{\infty}$ is unknown. *(i)* A dynamic scene in world coordinates. *(ii)* Direction of camera and car movement in world coordinates. *(iii)* Reference views as captured by the moving camera. *(iv)* Movement of objects under the fixed-camera formulation. *(v)* Assuming the background motion is purely vertical and the road is purely horizontal, linear interpolation of conjugate points on the car will produce a motion where the car stays on the same horizontal line on the interpolated road.

the y-axis, and the other is the original translation of the car along the road, which is entirely along the x-axis. When the car object is interpolated, the interpolation of the y-component of the car's fixed-camera motion will exactly match the interpolation of the background object (which is interpolated entirely in the y direction). Since the road is part of the background object, this means that the car will stay on the road at every step of the interpolation, even as it translates in the x direction. In particular, the wheels of the car will stay on the correct horizontal line of the road throughout the interpolation; if the wheels had been touching one of the straight edges of the road, then they would stay touching that edge (Fig. 1.3(v)).

Producing a view interpolation in which the car appears to move straight down the road is crucial for realism. The previous observation shows that this can be accomplished in some cases even without knowledge of $\mathbf{H}_{AB}^{\infty}$.

Is it necessary that the road be made purely horizontal and the background motion be made purely vertical before the interpolation is performed in order for the interpolated car object to stay on the road? The answer is "no" as we now demonstrate.

Assume the two reference views have been transformed so that (1) the image planes of both cameras are parallel to the common plane of the ground, and (2) the conjugate directions $\mathbf{u}_0$ and $\mathbf{u}_1$ are aligned in the views. At this point a transformation $\mathbf{M}$ that preserves z-coordinates (i.e., a 2D affine transformation) could be applied to both views so as to make the road parallel to the x-axis and the motion of the background object parallel to the y-axis. For instance, $\mathbf{M}$ could be a rotation followed by a shear, both in the x,y-plane.

By the discussion above, after $\mathbf{M}$ is applied the interpolated car object will always stay on the interpolated road. After the interpolation, $\mathbf{M}^{-1}$ could be applied to the virtual view to bring it into agreement with the reference views. The interpolated car will, of course, still be on the road after $\mathbf{M}^{-1}$ is applied. Mathematically, this process of interpolation is described by

$$\mathbf{M}^{-1}\left[(1-s)\frac{1}{\xi}\mathbf{M}\mathbf{H}_{UA}^{\infty}\mathbf{q}_U + s\frac{\lambda}{\xi}\mathbf{M}\mathbf{H}_{UB}^{\infty}(\mathbf{q}+\mathbf{u})_U\right] \tag{1.8}$$

The corresponding virtual camera is

$$\mathbf{M}^{-1}\left[(1-s)\mathbf{M}\mathbf{H}_{UA}^{\infty} + s\mathbf{M}\mathbf{H}_{UB}^{\infty}\right] \tag{1.9}$$

In both equations, the matrix $\mathbf{M}$ cancels out and the resulting equations exactly match those of Section 4.. The net effect is that it was not necessary to apply $\mathbf{M}$ at all; straight-forward interpolation would have kept the car on the road just as well.

Note that the previous discussion applies to any invertible $\mathbf{M}$ that preserves the z-component. This can be a powerful tool for generalizing view interpolation results. For instance, it could have been used to prove the results of Section 4..

5.4 AFFINE CAMERAS

Although the development for affine cameras is very similar to that for pinhole cameras, the underlying geometry of the two models is very different and requires a separate analysis.

The affine camera case is extremely important in practice. In addition, the algorithms for affine cameras are simpler and more robust than those for pinhole cameras.

Like a pinhole camera, an affine camera has an orientation in space and can be associated with a system of basis vectors. However, projection onto the image plane consists simply of setting the z-coordinate to 0. In effect, an affine camera has a z-axis corresponding to an infinitely long z basis vector. Using the fixed-camera formulation, in which the

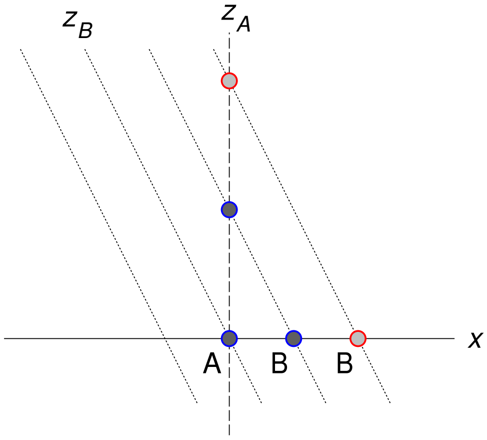

Figure 1.4 Two affine cameras share the same optical center, but not the same z-axis. The scene consists of two points lying on the z-axis of camera A. The points both project to the same location in camera A but to different locations in camera B, so no camera-to-camera transformation exists.

camera matrices are just 3×3, the third row of an affine camera is all 0's. Such a representation does not contain information about the camera's orientation and basis vectors. Hence, for the purposes of this section we will represent an affine camera A by the product of two matrices:

$$\widetilde{\mathbf{H}}^{\infty}_{UA} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \mathbf{H}^{\infty}_{UA} \tag{1.10}$$

Here $\widetilde{\mathbf{H}}^{\infty}_{UA}$ denotes the affine camera matrix for A while $\mathbf{H}^{\infty}_{UA}$ provides the underlying camera basis and orientation.

When cameras A and B share the same z-axis, there exists a camera-to-camera transformation. This transformation is given by the matrix

$$\widetilde{\mathbf{H}}^{\infty}_{AB} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \mathbf{H}^{\infty}_{AB} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{1.11}$$

When the camera-to-camera transformation $\widetilde{\mathbf{H}}^{\infty}_{AB}$ exists, resulting virtual views can be made to depict linear motion. $\widetilde{\mathbf{H}}^{\infty}_{AB}$ can be determined in this case from knowledge of just two conjugate directions: one conjugate direction is aligned with the x-axis of each view, the other with the y-axis, and then the x- and y-axes are scaled to place conjugate points on the same horizontal or vertical lines. Alternatively, a third conjugate

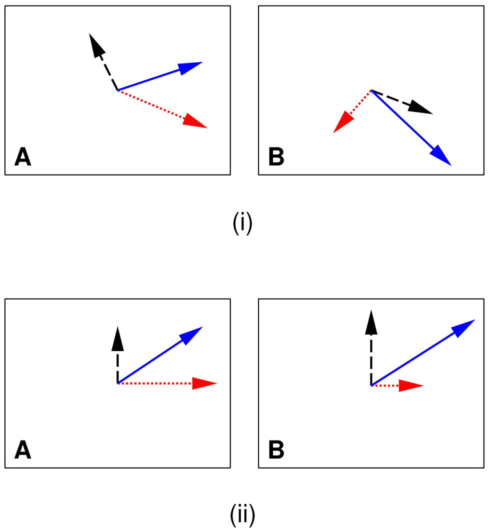

Figure 1.5 *(i)* Three conjugate directions as seen in view A and view B. *(ii)* It is possible to transform the views so that all three conjugate directions are aligned. In this case, a rotation is used to point the dotted vector along the x-axis, then a shear is used to point the dashed vector along the y-axis, and finally the x- and y-axes are scaled to align the solid vector.

direction could be used to determine scaling for the x- and y-axes (Fig. 1.5).

In general, except for when the z-axes are aligned, no camera-to-camera transformation exists for affine cameras (Fig. 1.4). The following results lead to a general-purpose algorithm for affine cameras:
Let Ω denote an object in the scene and let $\mathbf{u}$ be the associated translation vector. Let ω be a point belonging to the object and let $\mathbf{q}$ and $\mathbf{q}+\mathbf{u}$ denote the position of ω at times $t=0$ and $t=1$, respectively.

Projecting ω into camera A at time $t=0$ and into camera B at time $t=1$ and then linearly interpolating the projected image points yields

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} [(1-s)\mathbf{H}^{\infty}_{UA}\mathbf{q}_U + s\mathbf{H}^{\infty}_{UB}(\mathbf{q}+\mathbf{u})_U] \tag{1.12}$$

Because the x,y-projection matrix can be factored to the left, the development for affine cameras continues exactly as for pinhole cameras with one notable exception: There are no conditions on the reference views because the z-coordinates will always be 0, and consequently there is no prewarp step. As before, a virtual camera V exists such that the

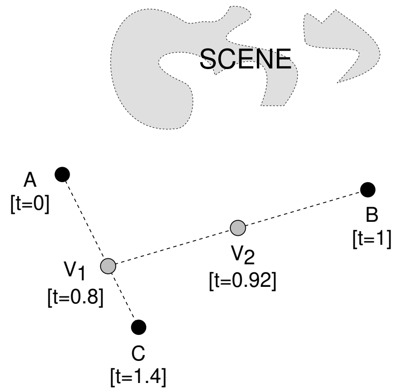

Figure 1.6 Virtual views can be created from multiple reference views by combining them in a pairwise manner. Here the reference views are A, B, and C, and the pairwise camera-to-camera transformations are known.

interpolation in Eq. 1.12 is the projection of scene point $\mathbf{q} + \mathbf{u}(s)$ into camera V. Camera V is defined by

$$\widetilde{\mathbf{H}}_{UV}^{\infty} = (1-s)\widetilde{\mathbf{H}}_{UA}^{\infty} + s\widetilde{\mathbf{H}}_{UB}^{\infty} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \mathbf{H}_{UV}^{\infty} \qquad (1.13)$$

and $\mathbf{u}(s)$ is as given in Eq. 1.6. In Eq. 1.6, $\mathbf{u}(s)$ is given in terms of basis V, which is the basis implicit in $\mathbf{H}_{UV}^{\infty}$. Note that the z-coordinate of $\mathbf{u}(s)$ is arbitrary.

The results of Section 4. pertaining to straight-line and linear virtual motion also hold for affine cameras.

It is always possible to simultaneously align three or fewer conjugate directions by using a technique analogous to that described above for finding $\widetilde{\mathbf{H}}_{AB}^{\infty}$ (Fig. 1.5). Any object whose conjugate translation direction can be aligned in both views will be portrayed as moving along a straight-line path in camera coordinates.

With pinhole cameras, the camera-to-camera transformation always exists so it is always possible to align any number of conjugate directions. With affine cameras, the camera-to-camera transformation may not exist so it is not always possible to portray straight-line motion for four or more objects.

6. INTERPOLATION WITH H^{∞}

If $\mathbf{H}_{AB}^{\infty}$ is known or can be determined, then view interpolations can be produced in a much more controlled fashion. As discussed in Section 4., an interpolation sequence can be created that portrays linear motion. Furthermore, as we now demonstrate, the space of possible intermediate

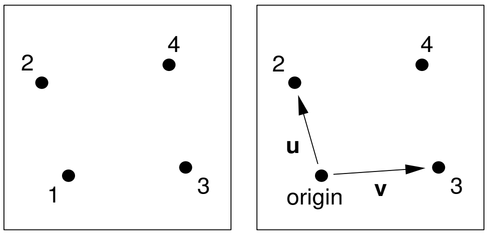

Figure 1.7 Reference view differences. The left-hand figure shows four points on a plane. By formally defining **u** to be the *difference* between point 2 and point 1, formally defining **v** to be the difference between point 3 and point 1, and formally taking point 1 to be the *origin*, point 4 can be represented as a linear combination of the quantities **u** and **v**. The same development exists if the points represent reference views acquired at various positions and times, assuming that linear motion can be achieved. Thus we can create a vector space of virtual views out of the purely formal concept of *reference view differences.* Note that each virtual view will have a time component as well as a space component.

virtual views that can be created from a set of reference views has an elegant vector-space structure.

So far, we have only presented view interpolation for two reference views. When three reference views are available, view interpolation can be applied in a pairwise manner to make use of all three reference views: An interpolation can be performed on two of the views, and the resulting virtual view can be interpolated with the third reference view. This process can be continued so as to incorporate an arbitrary number of reference views.

If the camera-to-camera transformation between each pair of reference views is known, the repeated pairwise interpolation will always portray linear virtual motion (Fig. 1.6). The order in which the pairwise interpolations are performed will not matter, and the final image will represent a "weighted sum" of reference-view differences analogous to a weighted sum of Euclidean vectors (Fig. 1.7).

A reference-view difference is just an abstract concept; it does not, for instance, mean subtraction of the intensity values of one image from another. The set of reference-view differences between one reference view and all the remaining views forms a basis for the vector space of all possible virtual views that can be created from the reference views. Therefore, just by considering such reference-view differences, we can quickly determine what virtual views can be created. For example, consider two reference views taken from the same location but at different times. There is only one reference-view difference in this case, so there is a one-dimensional space of possible virtual views. In particular, the

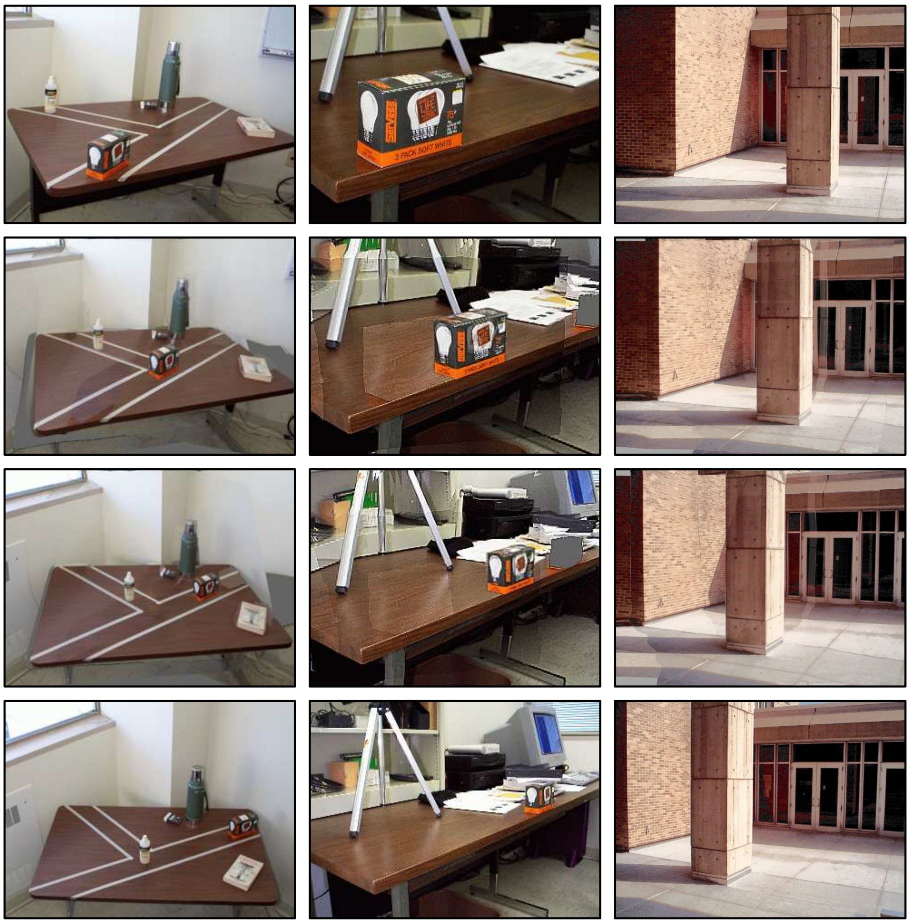

Figure 1.8 Experimental results.

view at any time from the given location can be generated. Two reference views taken at the same time that differ only in the x-coordinate of their positions can generate all views along the x-axis for that instant in time. To generate every possible view (every position and every time), five reference views are necessary (leading to a four-dimensional basis set).

Note that in general, except when linear motion can be achieved, the order in which the pairwise interpolations are performed will determine the final virtual view (i.e., the process is not commutative).

7. EXPERIMENTAL RESULTS

We have tested our methods on a variety of scenarios. Fig. 1.8 shows the results of three tests, each as a series of still images from a view interpolation sequence. The top-most and bottom-most images in each column are the two input reference views, and the center two images are virtual views created by the algorithm.

To create each sequence, two preprocessing steps were performed manually. First, the two reference views were divided into layers corresponding to distinct objects. Second, for each corresponding layer a set of conjugate points between the two views was determined. Since our implementation uses the Beier-Neely algorithm [3] for the morphing step, we actually determined a series of line-segment correspondences instead of point correspondences. For each sequence, between 30 and 50 line-segment correspondences were used (counting all layers).

For all the sequences, the camera calibration was completely unknown, the focal lengths were different, and the cameras were at different locations.

The first sequence is from a test involving three moving objects (including the background object). Since $\mathbf{H}_{AB}^{\infty}$ could only be approximated, the appearance of straight-line motion was achieved by aligning the conjugate directions of motion for each object during the prewarp step Section 5.3. An object's direction of motion is given by the epipoles of the object's fundamental matrix. Instead of calculating the fundamental matrices for the small objects on the table, we used the conjugate vanishing points of the tape "roads" for the same effect.

The second sequence involves two moving objects (counting the background object) and a large change in focal length. The third sequence demonstrates the parallel motion algorithm (Section 5.1). The scene is static, but the pillar in the foreground and the remaining background points are treated as two separate objects that are moving parallel to each other.

8. SUMMARY

We have demonstrated that physically-accurate, continuous interpolations between two reference views of a dynamic scene can be created in an image-based manner without reconstructing scene geometry. The method requires that, for each object in the scene, the movement that occurs between the two reference views must be equivalent to a rigid translation. For pinhole cameras, it also requires that, through a process of rectification, the image planes of both cameras can be made parallel to each other and to the direction of motion of any scene object for which the interpolation is desired. In some cases this can be achieved without determining the exact direction of motion of the scene objects. When the direction of motion of an object can be determined with respect to the two cameras' bases, the view interpolation will portray the object moving along a straight-line trajectory with respect to a camera-centered coordinate frame.

For affine cameras, no prewarping step is necessary; interpolation always produces a physically-correct interpolation. If the directions of motion of up to three scene objects can be determined, then a prewarp can be performed so that the view interpolation will portray straight-line motion with respect to a camera-centered coordinate frame. Since $\mathbf{H}_{AB}^{\infty}$ does not exist in general for affine views, the techniques presented in this chapter cannot be used to portray linear motion except in special cases.

For pinhole cameras, when $\mathbf{H}_{AB}^{\infty}$ can be determined, the interpolations can be made to portray linear motion. If more than two reference views are provided and if the camera-to-camera transformation can be determined between each pair of reference views, then the set of all possible virtual views that can be created is a vector space generated by reference-view differences.

Acknowledgments

The support of the National Science Foundation under Grant No. IIS-9530985, and the Defense Advanced Research Projects Agency and Rome Laboratory, Air Force Materiel Command, USAF, under Agreement No. F30602-97-1-0138 is gratefully acknowledged.

References

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

[2] S. Avidan and A. Shashua. Non-rigid parallax for 3D linear motion. In *Proc. Image Understanding Workshop*, pages 199–201, 1998.

[3] T. Beier and S. Neely. Feature-based image metamorphosis. In *Proc. SIGGRAPH 92*, pages 35–42, 1992.

[4] C. Bregler, M. Covell, and M. Slaney. Video rewrite: Driving visual specch with audio. In *Proc. SIGGRAPH 97*, pages 353–360, 1997.

[5] S. E. Chen and L. Williams. View interpolation for image synthesis. In *Proc. SIGGRAPH 93*, pages 279–288, 1993.

[6] J. C. Clarke and A. Zisserman. Detection and tracking of independent motion. *Image and Vision Computing*, 14:565–572, 1996.

[7] J. Davis. Mosaics of scenes with moving objects. In *Proc. Computer Vision and Pattern Recognition Conf.*, pages 354–360, 1998.

[8] G. J. Edwards, C. J. Taylor, and T. F. Cootes. Learning to identify and track faces in image sequences. In *Proc. Sixth Int. Conf. Computer Vision*, pages 317–322, 1998.

[9] O. D. Faugeras. What can be seen in three dimensions with an uncalibrated stereo rig. In *Proc. Second European Conf. Computer Vision*, pages 563–578, 1992.

[10] O. D. Faugeras, Q-T. Luong, and S. J. Maybank. Camera self-calibration: Theory and experiments. In *Proc. Second European Conf. Computer Vision*, pages 321–334, 1992.

[11] A. W. Fitzgibbon, G. Cross, and A. Zisserman. Automatic 3D model construction for turn-table sequences. In R. Koch and L. Van Gool, editors, *Proc. Workshop on 3D Structure from Multiple Images of Large-Scale Environments (SMILE '98)*, pages 155–170. Springer, 1998.

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

[13] R. I. Hartley. Self-calibration from multiple views with a rotating camera. In *Proc. Third European Conf. Computer Vision*, pages 471–478, 1994.

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

[15] M. Jagersand. Image based view synthesis of articulated agents. In *Proc. Computer Vision and Pattern Recognition Conf.*, pages 1047–1053, 1997.

[16] K. N. Kutulakos and S. M. Seitz. A theory of shape by space carving. In *Proc. Seventh Int. Conf. Computer Vision*, pages 307–314, 1999.

[17] R. A. Manning and C. R. Dyer. Interpolating view and scene motion by dynamic view morphing. In *Proc. Computer Vision and Pattern Recognition Conf.*, volume 1, pages 388–394, 1999.

[18] L. McMillan and G. Bishop. Plenoptic modeling. In *Proc. SIGGRAPH 95*, pages 39–46, 1995.

[19] M. Pollefeys. *Self-Calibration and Metric 3D Reconstruction from Uncalibrated Image Sequences.* PhD thesis, Katholieke Universiteit Leuven, Belgium, 1999.

[20] M. Pollefeys, R. Koch, and L. Van Gool. Self-calibration and metric reconstruction in spite of varying and unknown internal camera parameters. In *Proc. Sixth Int. Conf. Computer Vision*, pages 90–95, 1998.

[21] V. S. Ramachandran. Visual perception in people and machines. In A. Blake and T. Troscianko, editors, *AI and the Eye*, pages 21–77. Wiley, 1990.

[22] S. M. Seitz and C. R. Dyer. View morphing. In *Proc. SIGGRAPH 96*, pages 21–30, 1996.

[23] S. M. Seitz and C. R. Dyer. Photorealistic scene reconstruction by voxel coloring. *Int. J. Computer Vision*, 35(2):151–173, 1999.

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

[25] S. Ullman. *The Interpretation of Visual Motion.* MIT Press, Cambridge, Mass., 1979.

[26] Y. Yacoob and L. Davis. Computing spatio-temporal representations of human faces. In *Proc. Computer Vision and Pattern Recognition Conf.*, pages 70–75, 1994.