Interpolating View and Scene Motion by Dynamic View Morphing

Russell A. Manning **Charles R. Dyer***
Department of Computer Sciences
University of Wisconsin
Madison, WI 53706
{rmanning|dyer}@cs.wisc.edu
http://www.cs.wisc.edu/~{rmanning|dyer}

Abstract

We present a novel technique for interpolating between two views of a dynamic scene. Our approach extends the concept of view morphing introduced in [Seitz and Dyer, 1996] and retains the relative advantages of that method. The interpolation will portray *one possible* physically-valid version of what transpired in the scene during the intervening time between views. The scene is assumed to consist of a small number of *objects*. Each object can undergo any motion during the time between views as long as the total movement is equivalent to a single, rigid translation. The dynamic view morphing technique can work with widely-spaced reference views, sparse point correspondences, and uncalibrated cameras. When the *camera-to-camera transformation* can be determined, the virtual objects can be portrayed moving along straight-line, constant-velocity trajectories. Methods are developed for determining the camera-to-camera transformation from information available in the reference views. It is shown that each moving object in a scene has a corresponding fundamental matrix and that the camera-to-camera transformation can be determined from two distinct fundamental matrices. Dynamic view morphing is developed for both pinhole and orthographic cameras, and the use of three or more reference views is discussed. Static view morphing is made more versatile with respect to occlusion, and mosaicing is combined with dynamic view morphing for the case when both reference views share the same optical center. The resulting combination of techniques can be used to fill-in missing gaps in movies, perform "view hand-offs" between cameras at different locations, create movies from still images, perform movie stabilization and compression, track objects during periods of obstruction, and related tasks.

1 Introduction

This paper presents a novel method for interpolating between two views of a dynamic scene. View interpolation [Chen and Williams, 1993] is the process of creating a series of virtual views of a scene which, taken together, represent a continuous and physically accurate transition between two reference views of the scene. Most work on view interpolation is restricted to static scenes. Dynamic scenes change over time and, consequently, these changes will be evident in two reference views captured at different times. Therefore, view interpolation for dynamic scenes must portray a continuous change in viewpoint *and* a continuous change in the scene itself.

Our work is based upon an earlier technique called *view morphing* [Seitz and Dyer, 1996] which provides a method for interpolating between two widely-spaced views of a static scene (Fig. 1). The technique has several strengths which make it suitable for practical applica-

*This work is sponsored in part by the Defense Advanced Research Projects Agency (DARPA) and Rome Laboratory, Air Force Materiel Command, USAF, under agreement number F30602-97-1-0138.

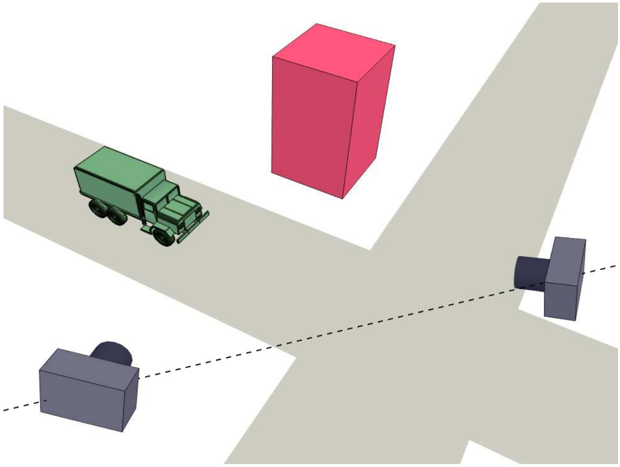

Figure 1: A static view morphing scenario. A building stands at the intersection of two roads, and a truck is parked near the building. The scene is viewed through two cameras. The view morphing algorithm can produce a series of new virtual views of the scene showing a gradual change in viewpoint between the two original camera views. If desired, the new viewing positions can be restricted to the line connecting the original cameras in space (the dotted line in the figure).

tions. First, only two reference views are assumed. Second, it does not require that camera calibration be provided nor does it need to calculate the camera parameters. Third, the method works even when only a sparse set of correspondences between the reference views is known. However, when more information about the reference views is known, this information can be used for added control over the output.

Because the original view morphing algorithm assumes a static scene, we refer to it as *static view morphing* to distinguish it from the *dynamic view morphing* technique presented in this paper.

We seek to perform view interpolation directly from the reference views, without additional information about the scene. Consequently, there will be a missing interval of time between when the reference views were captured, and it will be impossible to know for certain what occured during the missing interval. It is not our goal in this work to try and deduce the most likely manner in which the scene changed. Instead, we are interested in portraying *some* possible way in which the scene could have changed, and we want the portrayal to be physically accurate and continuous.

Our method is for dynamic 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.

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

A method for dynamic view interpolation, even if it is physically accurate, may be unsatisfactory if it portrays objects moving along unreasonable trajectories. For instance, when portraying a car driving along a bridge, it is essential that the car stay on the bridge during the entire sequence. To address this problem, we have developed techniques for portraying both straight-line motion (in a camera-based coordinate frame) and straight-line, constant-velocity motion (in both camera and world coordinate frames). We refer to the latter style of portrayal as *linear motion*.

If the reference cameras share the same position in world coordinates, then the virtual camera shares that position as well and straight-line motion relative to the virtual camera also implies straight-line motion in world coordinates. However, this may not be the case if the virtual camera moves during the view interpolation, as Fig. 3 demonstrates. If all objects can be portrayed undergoing linear motion in camera coordinates, then the virtual camera can be considered undergoing linear motion in world coordinates, in which case all the objects will undergo linear motion in world coordinates as well.

2 Static View Morphing

Static view morphing works by first prewarping the reference views to make their image

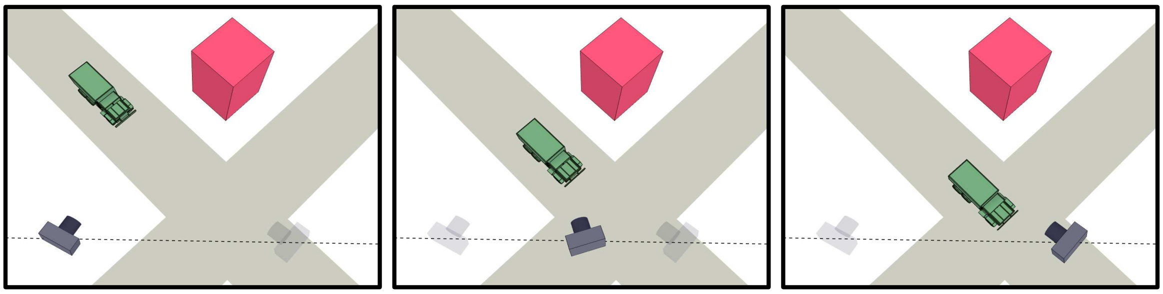

Figure 2: The scene in Fig. 1 becomes dynamic if the truck is driving down the road. The left and right frames above show the scene configuration when the reference views are taken at time $t = 0$ and $t = 1$, respectively. In both frames, the location of the corresponding reference camera is highlighted. The middle frame, which highlights the virtual camera, shows *one possible* recreation of the original scene at time $t = 0.6$. To recreate the scene, linear motion has been assumed (i.e., it has been assumed that the truck and the virtual camera both moved at constant velocity along straight-line paths). Dynamic view morphing produces a view of the recreated scene as captured by the virtual camera. The scene above has *two* moving objects: one is the truck and the other, called the "background object," consists of all stationary elements of the scene (such as the building and the road). The background object appears to move because the reference cameras are at different locations.

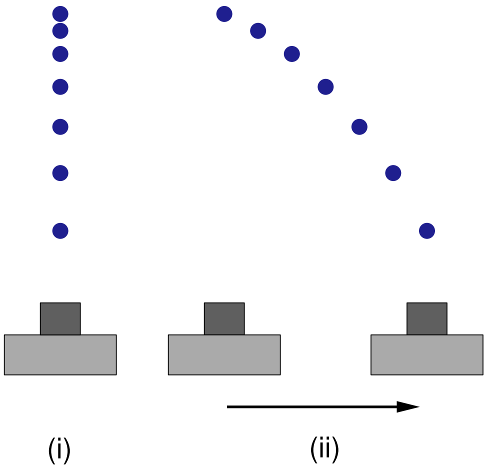

Figure 3: *(i)* A round object is filmed moving along a trajectory that is a straight line in the camera's frame of reference. The object is shown at equal time intervals and does not move at constant velocity. *(ii)* If the camera was in motion during the filming, then the object did not follow a straight-line trajectory in world coordinates.

planes parallel. After the prewarp, conjugate points in the two views are linearly interpolated to produce a physically accurate new view of the scene. The new locations of the conjugate points are used to guide a morphing algorithm in filling the remainder of the virtual view. Only the interpolated conjugate points are guaranteed to be viewed in the correct, physically accurate locations. By increasing the density of conjugate points, the virtual view can be made arbitrarily accurate. The prewarp is performed from information available in the fundamental matrix, which is calculated directly from the conjugate points. Complete details of the algorithm can be found in [Seitz and Dyer, 1996].

3 Dynamic View Morphing

3.1 Preliminary Concepts

We assume the two reference views are captured at time $t = 0$ and time $t = 1$ through pinhole cameras, which are denoted *camera A* and *camera B*, respectively.

We always use a *fixed-camera formulation*, meaning we assume that the two reference cameras are at the same location and that the world moves around them. Under this assumption, the camera matrices are just 3×3 and each camera is equivalent to a basis for $\Re^3$. We let U denote the "universal" or "world" coordinate frame, and use the notation $\mathcal{T}_{UA}$ to mean the transformation between basis U and basis A. Hence $\mathcal{T}_{UA}$ is the camera matrix for A. Of particular interest to our work is the matrix $\mathcal{T}_{AB}$. Note that capital script letters will always represent 3×3 matrices; in particular, $\mathcal{I}$ is the identity matrix.

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 cameras (that are not at the same location), then $\mathbf{e}_A$ is $\mathbf{e}$ measured in basis A. The quantity $\mathbf{e}$ is called the *epipole*. The fundamental matrix $\mathcal{F}$ for two cameras A and B that are at different locations has the following representation:

$$\mathcal{F} = [\mathbf{e}_B]_\times^+ \mathcal{T}_{AB} \tag{1}$$

Here $[\cdot]_\times^+$ denotes the transpose of 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 Ω is moving in direction $\mathbf{u}$, the fundamental matrix *for the object* is:

$$\mathcal{F}_\Omega = [\mathbf{u}_B]_\times^+ \mathcal{T}_{AB} \tag{2}$$

3.2 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 time $t=0$ and $t=1$, respectively (Fig. 4).

Assume 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 $\mathcal{T}_{UA}$ equals the third row of $\mathcal{T}_{UB}$ scaled by some constant λ. The second half means that $(\mathcal{T}_{UA}\mathbf{u}_U)_z = (\mathcal{T}_{UB}\mathbf{u}_U)_z = 0$, where $(\cdot)_z$ denotes the z-coordinate of a vector. Setting $Z = (\mathcal{T}_{UA}\mathbf{q}_U)_z = \lambda(\mathcal{T}_{UB}(\mathbf{q}+\mathbf{u})_U)_z$, the linear interpolation of the projection of ω into both cameras is

$$(1-s)\frac{1}{Z}\mathcal{T}_{UA}\mathbf{q}_U + s\frac{\lambda}{Z}\mathcal{T}_{UB}(\mathbf{q}+\mathbf{u})_U \tag{3}$$

Now define a virtual camera V by

$$\mathcal{T}_{UV} = (1-s)\mathcal{T}_{UA} + s\lambda\mathcal{T}_{UB} \tag{4}$$

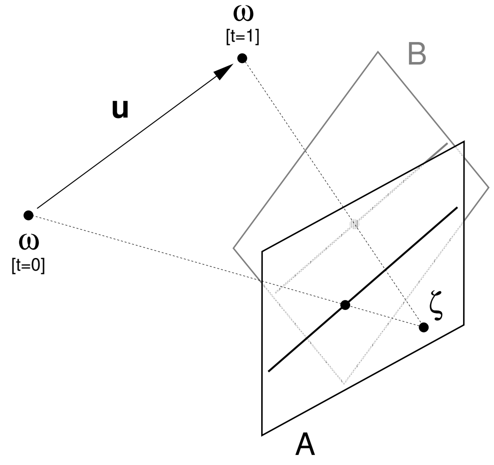

Figure 4: Cameras A and B share the same optical center ζ and are viewing a point on an object that translates by $\mathbf{u}$. The image planes of the cameras are parallel to each other and to $\mathbf{u}$, and hence interpolation will produce a physically correct view of the object. On each image plane a line parallel to $\mathbf{u}$ is shown.

Then the linear interpolation (3) is equal to the projection of scene point $\mathbf{q}(s)$ onto the image plane of camera V, where

$$\mathbf{q}(s) = \mathbf{q} + \mathbf{u}(s) \tag{5}$$
$$\mathbf{u}(s)_V = s\lambda\mathbf{u}_B \tag{6}$$

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 been translated by $\mathbf{u}(s)$ from its starting position.

Note that in Eq. 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. To have greater control over the interpolation process, we now prove that straight-line motion is achieved when $\mathbf{u}_A = \mathbf{u}_B$ up to an arbitrary scalar, and constant-velocity straight-line motion (i.e., linear motion) is achieved when $\mathbf{u}_A = \lambda\mathbf{u}_B$ (Fig. 5):

Assume $\mathbf{u}_A = k\mathbf{u}_B$ for some scalar k. Multiplying both sides of Eq. 4 on the right by $\mathcal{T}_{BU}$ yields

if prewarps make...	then interpolation is...
image planes parallel	physically correct
...*and* conjugate directions equal up to a scalar	...*and* depicts straight–line virtual motion
...*and* the scalar is λ	...*and* the motion is constant–velocity

Figure 5: How the virtual view of an object is related to different preconditions on the reference views. Stricter preconditions lead to increased control over the output.

$$\mathcal{T}_{BV} = (1-s)\mathcal{T}_{BA} + s\lambda \tag{7}$$

By multiplying both sides of Eq. 7 on the right by $\mathbf{u}_B$ and on the left by $\mathcal{T}_{VB}$ the following can be derived:

$$\mathcal{T}_{VB}\mathbf{u}_B = \frac{1}{(1-s)k + s\lambda}\mathbf{u}_B \tag{8}$$

Multiplying both sides of Eq. 6 by $\mathcal{T}_{VB}$ now yields:

$$\mathbf{u}(s)_B = \frac{s\lambda}{(1-s)k + s\lambda}\mathbf{u}_B \tag{9}$$

The basis V no longer plays a role and the virtual trajectory, given by $\mathbf{u}(s)_B$, is a straight-line in basis B. If $k = \lambda$ then $\mathbf{u}(s)_B = s\mathbf{u}_B$ and the virtual object moves at constant velocity. The results are in basis B, but multiplying by $\mathcal{T}_{BU}$ or $\mathcal{T}_{BA}$ indicates that the results also hold in world coordinates and camera A's coordinates, thus completing the proof. Keep in mind that the world coordinate system used in this context has its origin at the shared optical center of the reference cameras.

If $\mathcal{T}_{BA}$ is known then the camera matrix for B can be transformed into the camera matrix for A. This allows the view from camera B at time $t = 1$ to be transformed into the view from camera A at time $t = 1$, thus producing two views of the scene from camera A at different times. For this reason, we call $\mathcal{T}_{BA}$ the *camera-to-camera transformation*. By applying the earlier results to this special case, we derive the following corollary which forms the basis of the algorithm in Section 3.3:

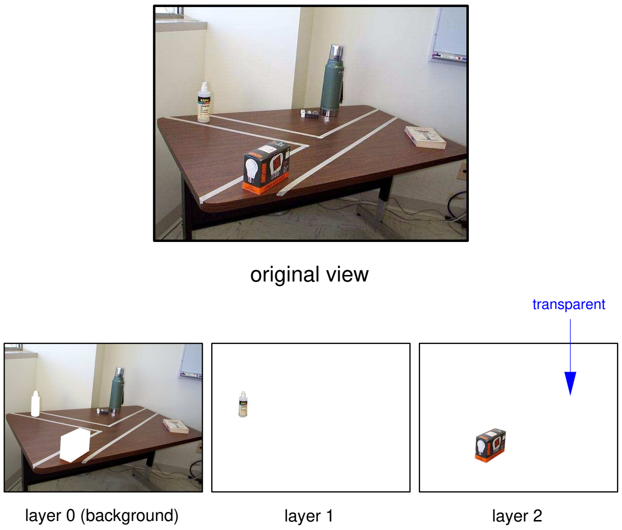

Figure 6: A view divided into layers. Each layer corresponds to a moving object. The single "background" object contains many different objects that all translate by the same amount.

If both camera matrices are equal and if $(\mathcal{T}_{UA}\mathbf{u})_z = 0$, *then the camera matrix for the virtual camera* V *is just* $\mathcal{T}_{UA}$ *and, because* $\lambda = 1$ *and* $\mathbf{u}_A = \mathbf{u}_B$, *the virtual object moves at constant velocity along a straight-line path.*

3.3 Linear Motion Dynamic View Morphing Algorithm

We now present a dynamic view morphing algorithm that will portray linear motion. The algorithm requires knowledge of $\mathcal{T}_{AB}$.

(Step 1) Segment both views into layers, with each layer representing a different moving object. Order the layers from nearest object to farthest object (Fig. 6).

(Step 2) Transform each layer of view B by $\mathcal{T}_{BA}$.

(Step 3) Apply static view morphing to each layer separately.

(Step 4) Recombine the new, virtual layers in the correct depth order.

(Step 5) (Optional) Postwarp the new view.

In step (3), the virtual camera will be the same for each layer by the corollary of the previous section. Furthermore, each layer will portray its corresponding object undergoing linear motion. Consequently, step (4) produces the desired linear portrayal of the entire scene.

3.4 Special Case: Parallel Motion

In this and the following section we examine some special case scenarios for which dynamic view interpolations can be produced without knowledge of $\mathcal{T}_{AB}$.

Assume a fixed-camera formulation and let $\mathbf{u}_i$ denote the 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 space.

Dynamic view morphing algorithm for 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 be parallel to the x-axis in both views; consequently, the virtual 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 moves parallel to the motion of all the objects and hence the virtual object motion is straight-line in world-coordinates.

3.5 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.

Recall that in Section 3.2 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 the objects simultaneously.

Dynamic view morphing algorithm for 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. *For each layer: prewarp the layer, linearly interpolate conjugate points, and morph the layer based on the interpolated conjugate points. Recomposite all the morphed layers to make a new, physically-accurate virtual view.*

The algorithm given above only guarantees physical correctness, not straight-line or linear virtual motion. Nevertheless, sequence (i) of Section 6, showing a box slide along the straight edge of a table, was generated using this algorithm. To make the sequence, the direction of the table edge was aligned in both views during the prewarp. The direction of the background object was also aligned. Why such alignment creates the desired effect is explained in [Manning and Dyer, 1998].

3.6 Orthographic Cameras

The mathematical development for orthographic cameras is similar to that for pinhole cameras. However, except in special cases, no camera-to-camera transformation exists between the reference cameras. Hence it is typically impossible to guarantee linear motion for the virtual objects. On the other hand, interpolation of conjugate points always produces a physically valid virtual view, without needing to make the image planes parallel. Prewarps can be applied to align conjugate directions and thus achieve straight-line motion. However, in general it is only possible to align at most three conjugate directions.

4 Finding $\mathcal{T}_{AB}$

The problem of determining $\mathcal{T}_{AB}$ is central to the linear motion algorithm of Section 3.3. $\mathcal{T}_{AB}$ can be determined from four conjugate directions by a well-known result used in mosaicing (because conjugate directions become conjugate points if we treat the reference cameras as being co-centered).

If the fundamental matrix can be determined for two objects in the scene and if the objects are not moving parallel to each other, then $\mathcal{T}_{AB}$ can be determined directly from these two fundamental matrices. The previous fact is proven in [Manning and Dyer, 1998], which also gives a method for approximating $\mathcal{T}_{AB}$ from two conjugate directions by making a reasonable assumption about the internal parameters of typical cameras.

5 Applications

Dynamic view morphing has many potential applications; we list a few here: filling a missing gap in a movie, creating a "hand-off" sequence to switch from one camera view to another, creating virtual views of a scene, removing obstructions or moving objects from a sequence, projecting motion into the future or past, stabilizing and compressing movie sequences, creating movies from still images, and using a single fixed camera to track moving objects.

6 Experimental Results

We have tested the concepts of this paper on a variety of scenarios. Fig. 7 shows the results of four tests, each as a series of still frames from a dynamic view interpolation sequence. The left-most and right-most frames of each strip are the original reference views, while the center two frames 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 the moving 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 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 every layer).

For all the sequences, the camera calibration was completely unknown and the focal lengths were different. Except in sequence (ii), the reference cameras were widely separated in space.

Sequence (i): Two moving objects, namely the background object and the box in the foreground. The sequence was created with the planar parallel motion algorithm.

Sequence (ii): Like sequence (i), but interpolating from a close-up view to a wide-angle view.

Sequence (iii): A scene with three-moving objects, which are the background and the two objects on the tape "roads." Created with the linear motion algorithm by approximating $\mathcal{T}_{AB}$.

Sequence (iv): A static scene with significant occlusion caused by the pillar. The sequence was created using the parallel motion algorithm by dividing the scene into two objects that move in the same direction (the pillar and everything else).

7 Conclusion

We have presented a method for interpolating between two views of a dynamic scene. The method requires that, for each object in the scene, the movement that occurs between the first view and the second must be equivalent to a rigid translation. The algorithm produces virtual views that portray one version of what might have occured in the scene. It is only necessary that the image planes of the reference cameras be parallel to each other and to the motion of an object for the interpolated view of the object to be physically accurate. With more conditions on the reference cameras, the virtual object can be portrayed moving along a straight-line path and even moving at constant velocity along a straight-line path. Interpolated views of a complete dynamic scene can be created by separately creating interpolated views of the scene's component objects and then combining the results.

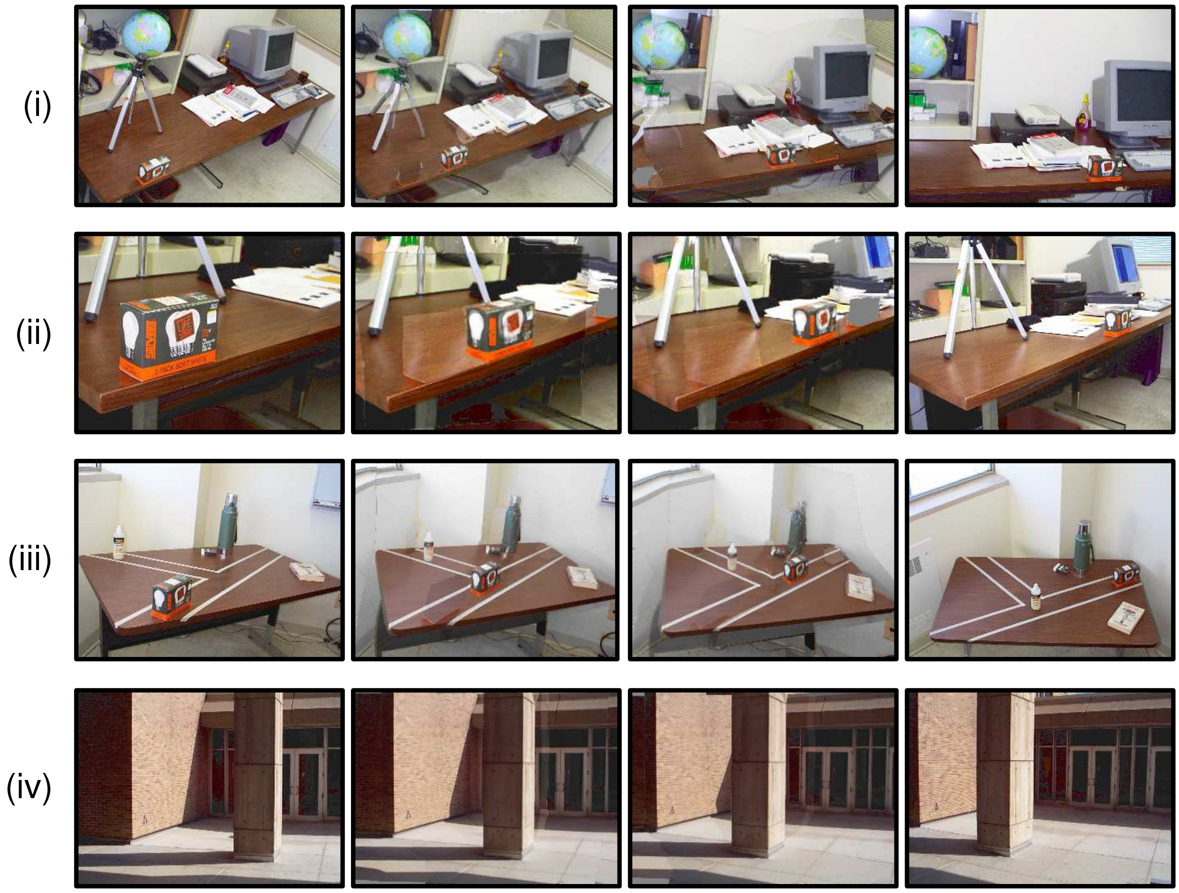

Figure 7: Experimental results.

By choosing to interpret the views as coming from the same position in space, a single theory has been created which applies to many different possible situations. In particular, the same theory applies whether or not the original reference cameras were actually co-centered. Since it is impossible to know from the reference views themselves how the original reference cameras were positioned relative to each other, the fixed-camera formulation is a natural default assumption. The virtual camera can be chosen to move along *any* trajectory; the choice simply alters the interpretation of the virtual views. The fixed-camera formulation also allows for a simple and intuitive development of the underlying mathematics of the theory.

Finally, it has been shown that each object in a dynamic scene has a corresponding fundamental matrix as long as the assumption of translational motion holds. From two distinct fundamental matrices, the camera-to-camera transformation can be determined.

The topics of this paper, as well as many additional topics and observations, are discussed in much greater detail in [Manning and Dyer, 1998].

References

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

[Manning and Dyer, 1998] R. A. Manning and C. R. Dyer. Dynamic view morphing. Technical Report 1387, Computer Sciences Department, University of Wisconsin-Madison, 1998.

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