Navigating Through Sparse Views

Shachar Fleishman
Tel Aviv University *

Baoquan Chen
SUNY at Stony Brook †

Arie Kaufman
SUNY at Stony Brook

Daniel Cohen-Or
Tel Aviv University

Abstract

This paper presents an image-based walkthrough technique where reference images are sparsely sampled along a path. The technique relies on a simple user interface for rapid modeling. Simple meshes are drawn to model and represent the underlying scene in each of the reference images. The meshes, consisting of only few polygons for each image are then registered by drawing a single line on each image to form an aligned 3D model. To synthesize a novel view, two nearby reference images are mapped back onto their models by projective texture-mapping. Since the simple meshes are a crude approximation to the real model in the scene, feature lines are used as aligning anchors to further register and blend two views together and form a final novel view. The simplicity of the modeling yields rapid, "home-made" image-based walkthroughs. We have produced walkthroughs from a set of photographs to show the effectiveness of the technique.

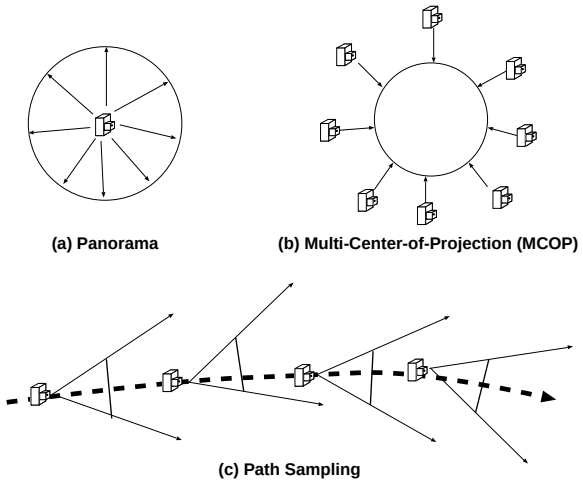


Figure 1: Camera sequences of different IBMR techniques.

1 Introduction

Creating animation or navigation from images has been an active subject for years. Image-based modeling and rendering (IBMR) is an emerging graphics research field of using images to model and render the scene [2, 3, 5, 8, 9]. The advantage of IBMR is that the tedious 3D modeling stage is avoided and traded by a number of precomputed images or photographs. On the other hand, since the complexity of image-based operations is independent of the scene complexity, rendering from images is usually faster than rendering complex geometric models.

The *plenoptic function* [9] represents the whole set of visible image information from a given viewing position. Interpolating a subset of sample views generates a novel view. Researchers have dedicated various ways of creating animation from a limited number of images. Each method has its constraints on camera configuration, thus offering different types of walkthroughs, which can be classified into three major categories:

Inside-out views:

In a panorama scene [2, 7, 12], photographs are taken from a fixed camera location, by rotating the camera $360°$. The photographs are then registered and mapped onto a cylindrical model forming a panoramic image (Figure 1a). From this panoramic image, rendering smooth walkthroughs into the scene is not straightforward.

Outside-in views:

Most of the image-based rendering techniques fall into this category. In the light field [8], lumigraph [5], and Multi-Center-Of-Projection (MCOP) [10] techniques, the modeling images are taken by placing the camera around the object of interest. The novel view is created by interpolation among these images. This implies that the walkthrough is restricted to looking at the object from the outside-in. Figure 1b sketches the camera configuration of MCOP techniques. However, the techniques in this category can be easily modified to support inside-out views.

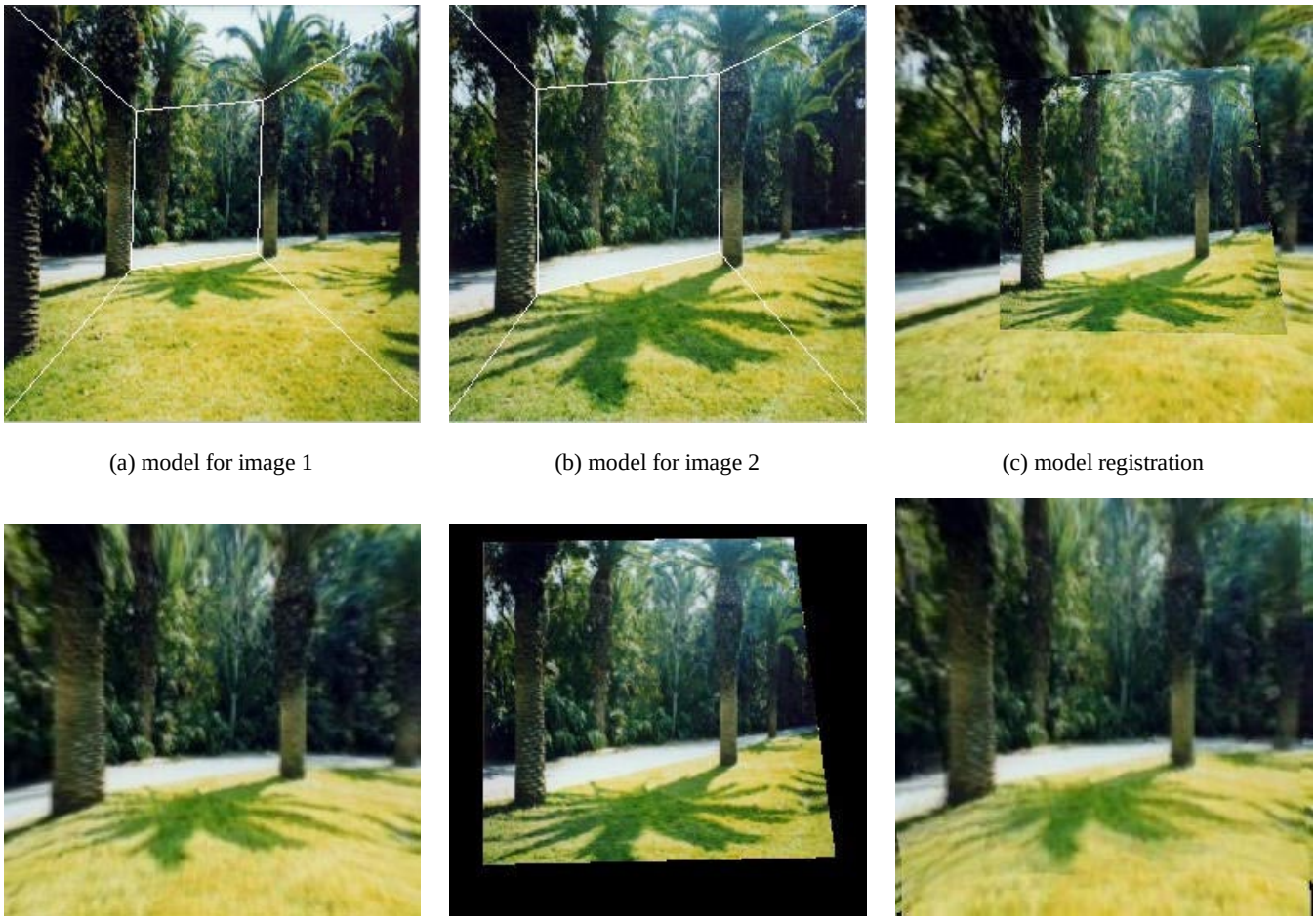
Walk-into views:

A recently developed technique called TIP (**T**our **I**nto the **P**icture) [6] provides a constrained navigation into a single image. In this technique, a simple underlying structure of the scene is constructed from a 2D *spidery mesh* drawn through user interface. The camera can then be shifted to a nearby location to render a novel view of the scene. This allows a virtual touring into the picture.

In this paper we present a technique to generate novel views from a sequence of reference images along a path through the scene (see Figure 1c). One camera is usually inside the previous camera viewing frustum, shifted further towards the scene. These sample views form the image-based model that represents the scene. The reference images are photographs taken by a regular camera without registering or calibrating the viewing positions. One immediate application for this technique is home entertainment. One can use a regular camera to take a sequence of shots when touring a landscape and reconstruct and view a continuous tour at a later time. It can also be regarded as a compact representation of a video sequence, which provide some degree of freedom for navigation.

Handling camera position changes, however, is considered a difficult problem [2]. Image morphing techniques (e.g., [1]) do not preserve any viewing transformation or rigid object transformation. The method proposed by Seitz et al. [11] provides viewing transformation from one image to another, but they do not handle the case where one camera is located inside the view frustum of another camera. Chen [2] briefly discussed this problem by creating multiple environment maps along the path. To guarantee a smooth transition, neighboring environment maps have to be placed very closely, which requires a large storage. Creating environment maps are tedious and the navigation path is still constrained. To allow navigation with more degrees of freedom, one has to construct accurate scene geometry, which necessarily involves an advanced modeling

*Email:{shacharf,daniel}@math.tau.ac.il
†Email:{baoquan,ari}@cs.sunysb.edu



(a) model for image 1 (b) model for image 2 (c) model registration

(d) novel view rendered from image 1 (e) novel view rendered from image 2 (f) final novel view by blending (d) and (e)

Figure 2: Algorithm Overview. (a) and (b) show the original input images and the models we used; (c) shows the registered models; (d) and (e) show a novel view from each model and (f) shows the final blended images.

interface and camera calibration. Successful techniques, e.g., in [4], exploit the constraints that are characteristic to architectural scenes.

The goal of our work is to develop a fairly simple technique by which an end-user can effectively create walkthroughs from a sequence of photographs. The TIP method has demonstrated that a simple underlying scene geometry is usually enough to effectively provide some degree of navigation into a single image. A striking feature of this technique is its simplicity in both modeling and rendering, and no requirement on camera calibration.

Here we introduce a method to generate a continuous walkthrough from sparse sample reference views. Our technique first uses an improved TIP technique to create a simple model for each reference image. The user also specifies matching feature lines between consecutive images. To create a novel in-between view, by utilizing the constructed models, we project the reference images to create two intermediate views. Next, the corresponding feature lines in the intermediate views are used to apply a morphing technique to blend the two intermediate views into the final novel view. In contrast to previous techniques, here the user defines independently "separate" models for each of the reference views. This significantly simplifies the modeling efforts, but of course the combination of the models yields noticeable deformations. However, these distortions are acceptable for a naive and fast modeling. This technique follows the spirit of "get 90% of the work in 10% of the effort".

2 Algorithm Overview

The input to our algorithm is a sequence of views along a path. To ease our explanation, we first discuss the case of only two input images. The extension to three and more images is straightforward. Given two nearby views sampled along a path, as in Figure 2, the algorithm consists of the following steps:

Step 1 Modeling: Independently, create a simple geometry for two images by employing a modified spidery mesh (Sec. 3).

Step 2 Model Registration: Register two geometries by placing the second geometry in the world space of the first geometry model (Sec. 4). This is achieved by computing two 3D-feature lines from user drawn matching feature lines in two images.

Step 3 Rendering and Blending: For each novel view, render two geometries by projective texture-mapping the original images onto the geometries. Meanwhile, project the feature

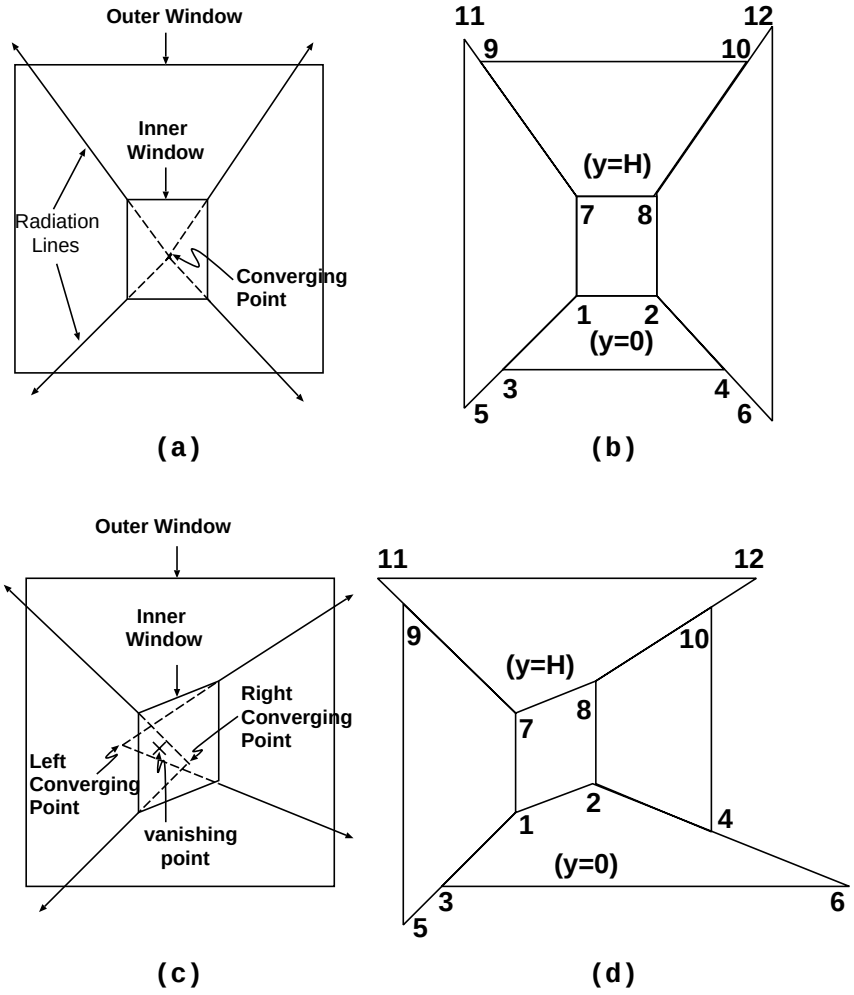


Figure 3: Original (a, b) and modified (c, d) spidery mesh.

lines in two intermediate views. Match corresponding feature lines of reference images by warping two intermediate novel views. Blend the two views together by a pixel-to-pixel cross-dissolve (Sec. 5).

Figure 2 illustrates a work flow of our algorithm.

3 Modeling

The original spidery mesh is too restrictive in describing many general scenes, since the 3D object represented by the spidery mesh is always a cubic 'room'. Figure 3a and b illustrate the original spidery mesh, consisting of floor and ceiling, left wall, right wall and rear wall, each is perpendicular to its adjacent wall/floor/ceiling. We relax some constraints on 2D mesh definition to achieve more flexible 3D modeling, however, without complicating the computation. The first relaxation is that the left wall and right wall do not have to be parallel to each other. This means that four horizontal lines of two walls do not necessarily converge to a single vanishing point. Therefore, in general, the left and right walls will have two separate converging points. The distance between the two converging points reflects the angle between two walls — the larger the distance, the greater the angle. The vanishing point now indicates the camera projection on the viewing plane. The second relaxation is that the rear wall does not have to be parallel to the viewing plane (i.e. perpendicular to the viewing direction). For a 2D mesh, this indicates that the rear wall (or inner window) can be any shape, but the intersections with the left and right walls have to be vertical. Figure 3c and d illustrate the modified mesh and its corresponding 3D model.

The computation of 3D vertices is still similar to [6]. We use a coordinate system in which x points to the right, y points up and z points to the camera with origin at the middle point of the bottom edge of image. We set the floor as $y = 0$. 3D coordinates of vertices 3 and 6 can be easily obtained. The vanishing point indicates the camera height above the floor. From an estimated viewing angle, we can obtain the distance to the viewing plane. The vector from the camera to vertex 1 on the viewing plane, called viewing vector of vertex 1, will intersect with the floor. By computing this intersection point, we obtain the 3D coordinates of vertex 1, similar to vertex 2. Knowing the x and z coordinate of vertex 7 (equal to vertex 1), by calculating the intersection between the vertical line (7,1) and the viewing vector for vertex 7, we get the height of edge (7,1). Similarly, we get the height for edge (8,2). We take the average of these two heights as the height for the ceiling. Now, the plane functions of left and right wall are known; therefore, we can calculate the coordinates for vertices 5, 9, 10, 4. From 9, 7, 8, we know the plane function of the ceiling, thus vertices 11, 12 can be calculated.

(a) input image (b) novel view 1 (c) novel view 2

Figure 4: Rendering using modified spidery mesh.

Figure 4 shows the results on an input image (Figure 4a) using the modified mesh. In Figure 4a, two converging points are stretched apart so that the modeled left and right walls match the underlying scene. Figure 4b and c present two rendered novel views.

4 Model Registration

Given two input image models, we need to position the second geometry in the 3D coordinate space of the first model. Illustrated in Figure 5, the modeling procedure now becomes:

Step 1 Model the black corridor (model A) for image 1, obtaining vertices $1 \sim 12$.

Step 2 Model the red corridor for image 2 (model B) , obtaining vertices $13 \sim 20$.

Step 3 Calculate the transformation of model B to the coordinate space of model A using the user drawn lines and apply the computed transformation to vertices $13 \sim 20$, unifying the coordinate spaces.

We assume that the floor of both the first geometry and the second geometry lies on the $y = 0$ plane. Therefore, to transform the second geometry to the 3D coordinate space of the first model, we need to find the orientation (y-rotation) and the depth position (z value) of the second geometry. We assume no z-rotation, though there is no theoretical difficulty in supporting it because it only involves a 2D image rotation. To achieve this, we let the user draw two points on the floor of both the first image and the second image. A third point can be used to determine the x-rotation.

Figure 6 illustrates a top view of registrating three image models. Matchine lines (also called registration lines) are drawn in consecutive images by the user. Using TIP-style models, 3D lines can be computed from these user drawn 2D lines on the floor. Given two lines like the magenta a, b line in Figure 6(a) and it's match in Figure 6(b) in two consecutive images, we can obtain a transformation matrix M by solving equation $M(a', b') = (a, b)$.

5 Image Blending

When creating novel views from a point between the COP (Center of Projection) of model A and the COP of model B, we need to blend the two in-between views so that a smooth transition from model A to model B is formed.

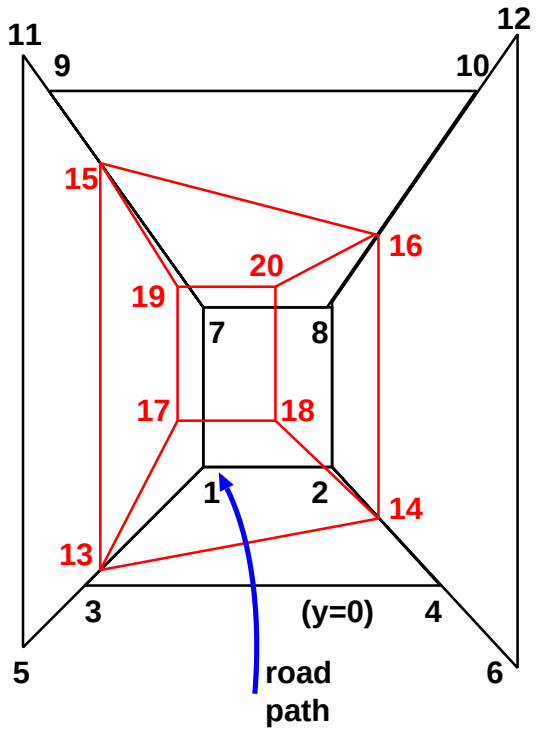


Figure 5: Registration of two image models.

The TIP modeling technique that we use is a simple modeling technique that gives pleasing visual results. Its disadvantage is that it is a very rough estimation of the underlying 3D geometry. Even though we have registered the two models, simply rendering two models and blending two intermediate images produce ghost-like effect as can be seen in Figure 7. To overcome this problem, the user registers matching features in the two input images, by drawing pairs of lines (also called feature lines), one on each reference image. To ease the drawing of matching line pairs, an edge-detection

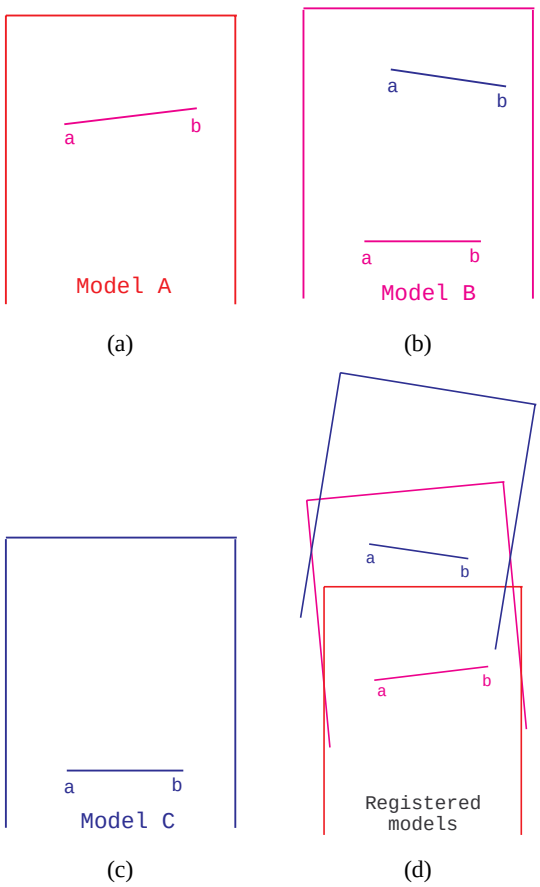


Figure 6: Top view of the registration of three images. (a) shows the model for the close (red) image; (b) shows the model for the middle (magenta) image; (c) shows the model for the far (blue) image. (d) shows the combined model where the magenta lines in models A and B overlap and the blue lines in models B and C overlap.

is applied to the two input images and the user simply mark pairs of corresponding lines.

To render the combined model, we render each 3D model from the novel viewpoint independently. Then we project the matching lines (feature lines) of the two models to find their 2D coordinates in the novel view and then we blend the two images using feature-based image morphing technique [1]. To simplify the explanation, we assume that the major movement direction is along the Z-axis, the time t of the morph is computed as $\frac{p_z - z_1}{z_2 - z_1}$, where z_1 is the z coordinate of the COP of the first model and z_2 is the z coordinate of the COP of the second model. In practice, we would like to make the transition time short, therefore, we use a non-linear function to represent blending weight for the second image, described as:

$$weight = \left\{ \begin{array}{ll} 0, & t < t_s \\ 1, & t > 1.0 \\ \frac{t - t_s}{1 - t_s}, & otherwise \end{array} \right\}$$, where t_s is the starting time of morph, e.g., 0.8.

6 Implementations and Results

The navigation system has two modes of operation; the first is the modeling mode where the user draws spidery meshes, registers them and draws pairs of matching lines. The second mode is the interactive walkthrough mode, where the user can navigate through

Figure 7: *Simple alpha blending of the two models results with ghost like effect, as can be seen in the trees or the road*

the images.

A typical modeling procedure takes the following steps using our user interface:

Model the first image, by drawing the back polygon and four rays that are originating from the corners of the back polygon. The four rays and the back polygon, define the floor, ceiling and the left and right walls.

Model the second image, using the same technique as the first image.

Model Registration. The user draws a line on the floor of the first image and a line on the second image, both lines are expected to overlap.

Feature registration. The user draws pairs of lines on important features of the scene.

The technique we have presented in this paper allows amateurs, home-users to create walkthroughs from a sequence of photographs. The user interface for the modeling and registration is simple enough so that a user can model a small sequence of images in a few minutes. The simplicity of the modeling produces crude 3D geometries, which prevents an accurate registration of multiple images. Applying standard image-space morphing techniques compensates for the ill modeling and registration.

Applying the navigation system is the only way to appreciate the results [1]. These movies demonstrate the transition through photographs, while navigating and changing the position and direction of the camera.

The navigation system is implemented on a PC-pentium platform with the OpenGL graphic library. It usually takes a few minutes for a user to prepare a navigation, i.e., drawing the model and featurelines. Rendering two intermediate views of 256×256 takes less than half a second, and the image morph takes about one seconds. The currently implementation is not optimized, and a significant acceleration is expected by optimizing the algorithms.

[1]The movies can be found on the project web-site at http://www.math.tau.ac.il/∼shacharf/NTSV/ntsv.html, or http://www.cs.sunysb.edu/∼baoquan/ntsv/index.html

One approach to improve the registration is to automatically reconstruct the 3D geometry of the scene using computer-vision techniques. Creating novel views from such models has the potential to produce accurate and high-quality images. However, these techniques tend to be insatiable and might produce errors in some cases, where as our technique gives up accuracy for guaranteed moderate results. Semi-automatic 3D reconstruction tends to be too involved and may consume a lot of time from an inexperienced user. Our approach takes the other extreme by providing a rapid prototype of the 3D model that produces reasonable visual effects. An interesting feature of our model is that opposed to traditional image-based rendering techniques, there are no holes in our system.

We are now considering exploring applications of the technique to movie sequences. The idea is to allow one to have freedom to navigate inside a movie while watching it. We are also interested in using the morph technique to register a sequence of images that are a combination of path sampling and panorama. This will add more flexibility to the technique, which will expand the range of images the technique can be applied to.

Acknowledgments

This work was supported by a grant from the Israeli Ministry of Science and by ONR grant N000149710402 and NSF grant MIP9527694.

References

[1] Thaddeus Beier and Shawn Neely. Feature-based image metamorphosis. *Computer Graphics (SIGGRAPH '92 Proceedings)*, 26(2):35–42, July 1992.

[2] S. E. Chen. Quicktime VR - an image-based approach to virtual environment navigation. In *SIGGRAPH 95 Conference Proceedings*, pages 29–38, August 1995.

[3] S. E. Chen and L. Williams. View interpolation for image synthesis. In *SIGGRAPH '93 Conference Proceedings*, volume 27, pages 279–288, August 1993.

[4] Paul E. Debevec, Yizhou Yu, and George Borshukov. Efficient view-dependent image-based rendering with projective texture-mapping. *Rendering Techniques'98*, pages 105–116, 1998.

[5] S. J. Gortler, R. G., R. Szeliski, and M. F. Cohen. The lumigraph. In *SIGGRAPH 96 Conference Proceedings*, pages 43–54, August 1996.

[6] Y. Horry, K.I Anjyo, and K. Arai. Tour into the image: Using a spidery mesh interface to make animation from a single image. In *SIGGRAPH 97 Conference Proceedings*, pages 225–232, August 1997.

[7] Sing Bing Kang and Pavan Kumar Desikan. Virtual navigation of complex scenes using clusters of cylindrical panoramic images. In *Graphics Interface*, pages 223–232, June 1998.

[8] M. Levoy and P. Hanrahan. Light field rendering. In *SIGGRAPH 96 Conference Proceedings*, pages 31–42, August 1996.

[9] L. McMillan and G. Bishop. Plenoptic modeling: An image-based rendering system. In *SIGGRAPH 95 Conference Proceedings*, pages 39–46, August 1995.

[10] Paul Rademacher and Gary Bishop. Multiple-center-of-projection images. In *SIGGRAPH 98 Conference proceedings*, pages 199–206, August 1998.

[11] S. M. Seitz and C.R. Dyer. View morphing. In *SIGGRAPH 96 Conference Proceedings*, pages 21–30, August 1996.

[12] Richard Szeliski and Heung-Yeung Shum. Creating full view panoramic mosaics and environment maps. In *SIGGRAPH 97 Conference Proceedings*, pages 251–258, August 1997.