12-1-2004

# Creating walk-through images from a video sequence of a dynamic scene

Hyung W. Kang
*University of Missouri-St. Louis*, kangh@umsl.edu

Sung Yong Shin
*Korea Advanced Institute of Science & Technology*

December, 2004

# Creating Walk-Through Images from a Video Sequence of a Dynamic Scene

Henry Kang, *University of Missouri-St. Louis*
Sung Y Shin

**Hyung W. Kang**
Computer Science Division
University of Missouri–St. Louis
8001 Natural Bridge Rd.
St. Louis, MO 63121 USA
kang@cs.umsl.edu

**Sung Yong Shin**
Korea Advanced Institute of
Science and Technology
Taejon 305-701 South Korea

# Creating Walk-Through Images from a Video Sequence of a Dynamic Scene

## Abstract

Tour into the picture (TIP), proposed by Horry et al. (Horry, Anjyo, & Arai, 1997, *ACM SIGGRAPH '97 Conference Proceedings,* 225–232) is a method for generating a sequence of walk-through images from a single reference image. By navigating a 3D scene model constructed from the image, TIP provides convincing 3D effects. This paper presents a comprehensive scheme for creating walk-through images from a video sequence by generalizing the idea of TIP. To address various problems in dealing with a video sequence rather than a single image, the proposed scheme is designed to have the following features: First, it incorporates a new modeling scheme based on a vanishing circle identified in the video, assuming that the input video contains a negligible amount of motion parallax effects and that dynamic objects move on a flat terrain. Second, we propose a novel scheme for automatic background detection from the video, based on 4-parameter motion model and statistical background color estimation. Third, to assist the extraction of static or dynamic foreground objects from video, we devised a semiautomatic boundary-segmentation scheme based on enhanced lane (Kang & Shin, 2002, *Graphical Models, 64*(5), 282–303). The purpose of this work is to let users experience the feel of navigating into a video sequence with their own interpretation and imagination about a given scene. The proposed scheme covers various types of video films of dynamic scenes, such as sports coverage, cartoon animation, and movie films, in which objects are continuously changing their shapes and locations. It can also be used to produce a variety of synthetic video sequences by importing and merging dynamic foreign objects with the original video.

## 1 Introduction

### 1.1 Motivation

Real-time generation of photorealistic images is a recurring theme in computer graphics. Recently, a novel approach for real-time realistic image generation called image-based rendering has received much attention. Tour into the picture (TIP), proposed by Horry, Anjyo, and Arai (1997), is one of the image-based methods for generating a sequence of walk-through images from a single reference image. By navigating a 3D scene model constructed from the image, TIP provides convincing 3D effects. Assuming that the image has one vanishing point, Horry et al. proposed a scene-modeling scheme called "spidery mesh," with which users can make what they imagine in the 2D image become real in 3D.

Presence, *Vol. 13, No. 6, December 2004,* 638–655

Due to widely available video imaging devices such as camcorders or CCTVs and the growth of the Internet, which abounds with digitized movie files, video clips are now emerging as familiar sources for image-based techniques. This trend motivates the need for a more general version of an image-based navigating scheme that can deal with video sequences. However, we cannot directly apply the original TIP to video sequences because TIP is designed to handle just a still image. While various image-based techniques have been proposed, most of them use a set of still photographs (Chen & Williams, 1993; Chen, 1995; McMillan & Bishop, 1995; Gortler, Grzeszczuk, Szeliski, & Cohen, 1996; Levoy & Hanrahan, 1996) rather than video sequences. Moreover, previous work on video sequences usually focused on automatic reconstruction of structures from static scenes in which only the camera moves around fixed objects (Tomasi & Kanade, 1992; Beardsley, Torr, & Zisserman, 1996; Fitzgibbon & Zisserman, 1998).

This paper presents a scheme for creating walk-through images from videos by generalizing the idea of TIP. To address various problems in dealing with a video sequence rather than a single image, the proposed scheme is designed to have the following features: First, it adopts a new modeling scheme based on the notion of a *vanishing circle,* which is more general and simpler than that of TIP. Second, we propose a novel scheme for automatic background detection from the video taken with camera rotation and zoom. Third, for efficient extraction of static or dynamic foreground objects from the video, we present a semiautomatic boundary-segmentation scheme based on *enhanced lane* (Kang & Shin, 2002a).

The proposed scheme aims at helping users experience the feel of navigating into the video sequence with their own interpretation of the scene, and create new synthetic videos by importing and compositing foreign objects according to their own imaginations. Our scheme covers various types of video films of dynamic scenes such as TV broadcast, cartoon animation, and movie films, where objects are allowed to change their shapes and locations continuously.

### 1.2 Related Work

McMillan and Bishop (1995) explained image-based rendering with the notion of a plenoptic function, which defines the radiant energy to an eye position through every incident direction. For example, an environment map is a sample of a plenoptic function at a fixed viewpoint (Lippman, 1980; Miller et al., 1992; Chen, 1995). An image-based rendering scheme based on environment maps has a major limitation in that the viewpoint is fixed. One way to relax this limitation is to use a warp function that describes the relative movement of each pixel with respect to camera movement (Chen & Williams, 1993; Darsa, Silva, & Varshney, 1995; McMillan & Bishop). Alternatives are to construct a light field from a set of plenoptic samples such as multiple reference images taken at regular grid points (Gortler et al., 1996; Levoy & Hanrahan, 1996; Sloan, Cohen, & Gortler, 1997).

TIP, proposed by Horry et al. (1997), generates realistic walk-through images by constructing a simple 3D scene model from a 2D image. However, with the assumption that the image has a single vanishing point, their modeling scheme requires major modification when the image contains multiple vanishing points or no clearly identified vanishing point. Liebowitz, Criminisi, & Zisserman (1999) presented algorithms for computing plane rectification or plane orientation to reconstruct architectural models from a single image, exploiting various geometric constraints such as parallelism and orthogonality.

Recently, Kang, Pyo, Anjyo, & Shin (2001) proposed a new modeling scheme for TIP based on a vanishing line that is simpler than that of Horry et al., and yet more general to cover a broader class of input images. They also showed that their modeling scheme is naturally extended to navigation into a panoramic image, by introducing the notion of a vanishing circle. Compared to conventional panoramic-image viewers (such as QuickTimeVR®), their method can provide the real sense of walk-through or navigation into the panoramic scene by enabling continuous camera translation as well as rotation. In this paper, their modeling scheme for a

single planar or panoramic image is further extended to a video sequence.

The image-based rendering techniques described above are common in that they all use one or more static images to obtain 3D scene information for generating an image viewed from a new viewpoint. On the other hand, some researchers, especially in the field of computer vision, have concentrated on automatically extracting 3D information from video sequences (Tomasi & Kanade, 1992; Beardsley et al., 1996; Fitzgibbon & Zisserman, 1998). These approaches have usually focused on static scenes, which contain relatively simple architectural models such as buildings or houses. Thus, they cannot be applied to objects that are moving arbitrarily or changing their shapes continuously, for example, pedestrians and soccer players. Thus, dynamic scenes of this kind have usually been the target of 2D motion tracking or segmentation (Mitsunaga, Yokoyama, & Totsuka, 1995; Vieren, Cabastaing, Postaire, 1995; Hoch & Litwinowicz, 1996; Wren, Azarbayejani, Darrell, & Pentland, 1997), rather than that of 3D model reconstruction.

### 1.3 Overview

In this paper, we present an image-based navigation scheme for video sequences of dynamic scenes. While a previous version of our work has been published (Kang & Shin, 2002b), this paper provides a more detailed and extended description of this work. Our scheme is based on the following assumptions: First, the input video is composed of a continuous sequence of images for a scene. Second, only a negligible amount of motion parallax effects appear in the video. Third, the terrain on the ground (which appears in the video) is smooth enough so that it can be modeled as a single plane. In general, most of the video sequences containing dynamic scenes satisfy these assumptions, that is, dynamic objects seldom move on a rough terrain, and it is hard to make a large amount of camera translation (which causes a strong parallax effect) while tracking the moving objects with the video camera at the same time.

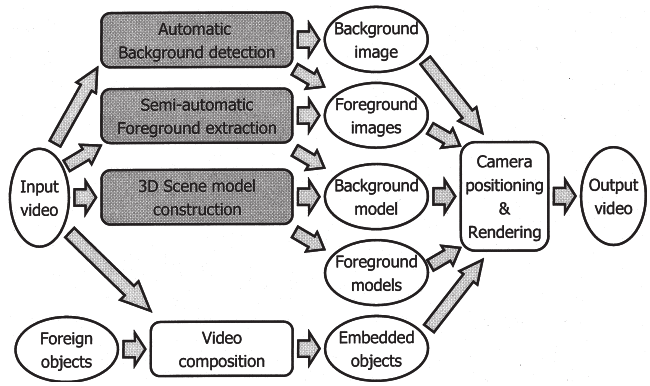


**Figure 1.** *Schematic of TIV.*

Figure 1 shows the process flow diagram of our tour into video (TIV) scheme. First, a single background image is generated from an input video sequence. The background image covers all the region viewed from the entire sequence of frames, and contains only the static entities in the scene, that is, the background and static foreground objects. We generate the background image by employing an automatic background-detection technique (Francois & Medioni, 1999; Stauffer & Grimson, 1999) in conjunction with an image-alignment (registration) algorithm based on a 4-parameter motion model to compute a camera pose for each frame (Shum & Szeliski, 1998).

For each static foreground object, a corresponding region is interactively extracted from the background image, for which we use a highly interactive image-segmentation tool called "enhanced lane" (Kang & Shin, 2002a). For dynamic foreground objects, regions should be identified in each frame. With the camera poses and the background image obtained from the background detection process, we can extract the boundary information of dynamic foreground objects in each frame by applying a connected component algorithm on a difference image between each frame and its background image (Horn, 1986). For frames with incorrect segmentation results due to noises or ambiguities, we go through an iterative segmentation process to correct the given boundary from frame to frame. This iterative process is composed of three steps, including

enhanced lane, block matching, and active contour (Kass, Witkin, & Terzopoulos, 1987).

Given the background and the foreground information thus extracted, we construct the 3D scene model, which consists of a background model and foreground models. The modeling scheme for video sequences presented here evolved from that for a single image, in that it receives a single background image as an input. The background model is first constructed based on a vanishing circle detected in the background image. For static or dynamic foreground objects, we first place 2D polygons bounding the extracted objects in the background image or the reference frame. They are then modeled as polygons in 3D space and attached to the background model after their 3D coordinates are computed. The regions inside their corresponding 2D polygons serve as their foreground texture maps (called "foreground image") where only the exact portion of each object is marked as visible.

With the constructed scene model and all the texture images prepared, the dynamic scene can be navigated by positioning the camera and successively creating images viewed from new viewpoints. Note that the polygon for each dynamic foreground object is continuously changing its shape and location on the scene model from frame to frame. It is also possible to create a new synthetic video sequence by importing foreign objects, either static or dynamic, into the scene. Because all the foreground objects are modeled as polygons with textures, even complex objects with arbitrary colors can be inserted to enrich the virtual environment. The capability for generating augmented reality (AR) of this type can be used for postproduction in the film industry (Milgram, Shumin, Drascic, & Grodski, 1993; Azuma, 1997; Azuma et al., 2001).

The remainder of this paper is organized as follows: In Section 2, we present an automatic background-image generation method. Section 3 discusses the semi-automatic boundary-segmentation process, with which static or dynamic foreground objects can be effectively extracted from a video sequence. In Section 4, the construction scheme for the background model and the foreground object models is described in detail. Section 5 provides some experimental results with example video sequences. Finally, we conclude this paper and suggest some future extensions in Section 6.

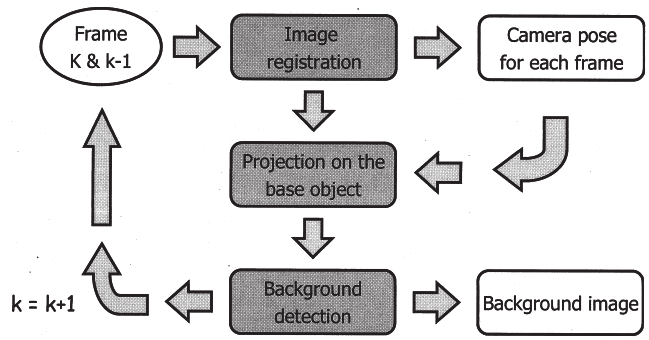


**Figure 2.** *Block diagram for background-image generation.*

## 2 Background-Image Generation

To construct a single background image from an input video sequence, successive frames in the sequence should be aligned first. The image-alignment (or registration) algorithm in this work uses a 4-parameter motion model that can handle camera rotation and zoom (Shum & Szeliski, 1998). Each registered frame is then projected onto a spherical base object to generate a single image. An appropriate color value is assigned to each pixel of the resulting image so that the pure background information remains with all the dynamic foreground objects removed. This process is called background detection, for which we adopt a pixel-based adaptive, statistical model (Francois & Medioni, 1999; Stauffer & Grimson, 1999). Figure 2 shows the block diagram for our background-image generation process.

### 2.1 Image Registration

For aligning frames in the video sequence, we employ the 4-parameter motion model proposed by Shum & Szeliski (1998), which incorporates both camera rotation and zoom. Compared to the traditional 8-parameter motion model (Bergen, Anandan, Hanna, & Hingorani, 1992), this model provides faster and more robust

convergence to the aligned position. Another benefit of this model is that it explicitly computes the camera pose for each frame, which is essential for generating a background image on a base object other than a plane. As mentioned in the previous section, we assume that motion parallax effects are negligible in the input video. That is, the factor of camera translation can be ignored so that we can fix the camera position at a single point in 3D.

When two images are taken from the same viewpoint but in different directions, the relationship between the two images can be described by a planar homography (Hartley & Zisserman, 2000). Thus, one image is warped into another using a $3 \times 3$ matrix $\mathbf{H}$ as $\mathbf{x}' \sim \mathbf{H}\mathbf{x}$, where $\mathbf{x} = (x, y, 1)$ and $\mathbf{x}' = (x', y', 1)$ are homogeneous coordinates, and $\sim$ indicates equality up to scale. For a camera centered at the origin, the relationship between an image point x and its corresponding 3D point $\mathbf{p} = (X, Y, Z)$ can be described by $\mathbf{x} \sim \mathbf{K}\mathbf{R}\mathbf{p}$, where $\mathbf{K}$ and $\mathbf{R}$ are a simplified camera calibration matrix and a 3D rotation matrix, respectively. Without loss of generality, we assume that the origin of the pixel coordinate is at the image center. The planar homography $\mathbf{H}$ between two frames $k$ and $k - 1$ is then given by

$$\mathbf{H} \sim \mathbf{K}_k\mathbf{R}_k\mathbf{R}_{k-1}^{-1}\mathbf{K}_{k-1}^{-1} \tag{1}$$

where $\mathbf{K}_k$ and $\mathbf{R}_k$ respectively denote the camera calibration matrix and the rotation matrix for frame $k$. The rotation matrix can be recovered by incrementally updating $\mathbf{R}_k$ based on the angular velocity $(\omega_x, \omega_y, \omega_z)$, and a focal length $f_k$ of $\mathbf{K}_k$ can be adjusted by setting $f_k \leftarrow (1 + e_k)f_k$, where $e_k$ is the incremental change of the focal length.

We initially place the current frame $k$ and the previous frame $k - 1$ at the same position. To align the two frames, we find $\mathbf{q} = (\omega_x, \omega_y, \omega_z, e_k)$, which minimizes the squared error metric

$$E(\mathbf{q}) = \sum_i [I_k(\mathbf{x}_i') - I_{k-1}(\mathbf{x}_i)]^2 \tag{2}$$

The least squares problem given by Equation 2 can be solved through the standard procedure in Press, Flannery, Teukolsky, and Vetterling (1992). By minimizing $E(\mathbf{q})$, we estimate the incremental rotation vector $(\omega_x, \omega_y, \omega_z)$ and the incremental change of the focal length $e_k$, after which $\mathbf{R}_k$ and $\mathbf{K}_k$ can be updated.

### 2.2 Projection on a Base Object

After the camera pose for the current frame is obtained by the image-registration process, the frame is projected onto a base object to create a single background image. While there can be various candidates for the base object, including a plane, a cylinder, a sphere, and a cube (Chen, 1995), in this paper we focus mainly on the sphere, as it is the most general type. For example, a spherical base object can deal with the entire viewing range covered by the video sequence even if it covers more than 180° in the horizontal direction, and more than 90° in the vertical direction.

The projective mapping is done by using the camera pose information (rotation matrix $R_k$ and focal length $f_k$). For example, we can construct a spherical background image by first converting each pixel $\hat{\mathbf{x}} = (\theta, \phi)$ on this image into its corresponding 3D direction vector $\mathbf{p} = (\cos\theta\cos\phi, \sin\theta\cos\phi, \sin\phi)$, and then determining its mapping onto each frame $k$ using $\mathbf{x} \sim \mathbf{K}_k\mathbf{R}_k\mathbf{p}$ and assigning an appropriate pixel color to form an updated spherical background image (Figure 3a).[1]

In case we choose a cylinder as a base object, each pixel $\hat{\mathbf{x}} = (\theta, z)$ on the cylindrical background image is converted to the 3D direction vector $\mathbf{p} = (\cos\theta, \sin\theta, z)$ to get the corresponding pixel x on each frame (Figure 3b).

### 2.3 Background Detection

While projecting each frame on the base object, each pixel on the constructed image should be updated with an appropriate color value. To obtain a pure background image from a sequence of images, we adopt a pixel-based adaptive, statistical background-detection method based on a Gaussian mixture model (Francois

[1]Similarly, $\hat{\mathbf{x}} = (\theta, \phi)$ can be obtained from point $\mathbf{x}$ on frame k using $\mathbf{p} \sim \mathbf{R}_k^{-1}\mathbf{K}_k^{-1}\mathbf{x}$ and finding the intersection between $\mathbf{p}$ and the base sphere.

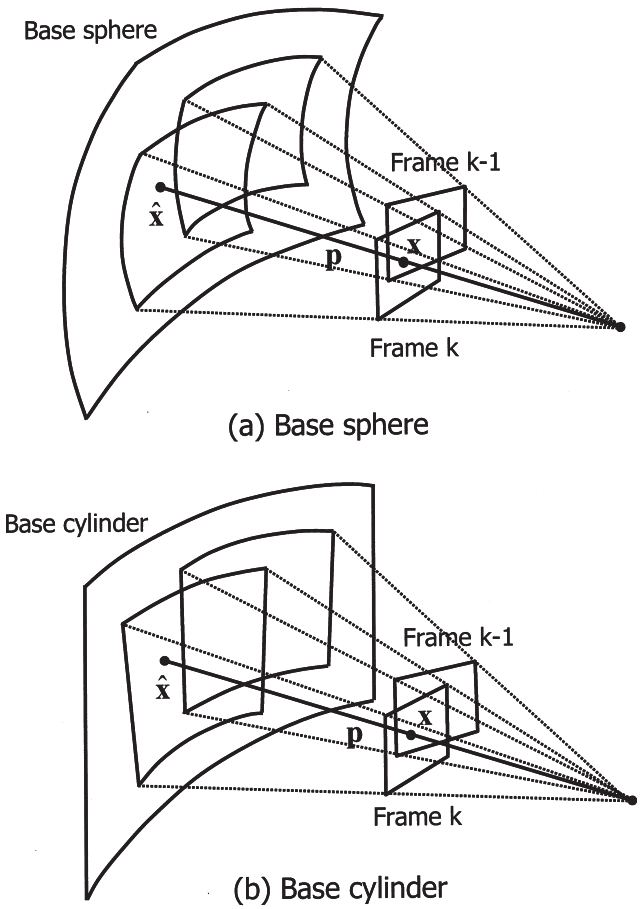


**Figure 3.** *Projection on a base object.*

& Medioni, 1999; Stauffer & Grimson, 1999). Compared to previous approaches for statistical background detection (Koller et al., 1994; Friedman & Russell, 1997), this model is more effective in dealing with scenes where the background color of each pixel may dynamically change by object movements, shadows, etc. The intensity values of a pixel over time are considered as a stochastic process, which means a time series of pixel values. For each pixel $\hat{\mathbf{x}}$ on the background image $I_B$, we keep track of its history

$$\{z_i : z_i = I_B(\hat{\mathbf{x}}, i), 1 \leq i \leq k\} \tag{3}$$

where $I_B(\hat{\mathbf{x}}, i)$ is the intensity value at $\hat{\mathbf{x}}$ at the $i$th frame in the sequence. The recent history of each pixel is modeled by a mixture of $N$ Gaussian distributions and the probability of observing the current pixel value is

$$P(z_k) = \sum_{i=1}^{N} \omega_{i,k}\eta(z_k, \mu_{i,k}, \sigma_{i,k}^2) \tag{4}$$

where $N$ is the number of distributions, $\omega_{i,k}$ is an estimate of the weight (indicating the number of occurences of the intensity value that is accounted for by this Gaussian) of the $i$th Gaussian in the mixture at time $k$, and where $\eta$ is the corresponding Gaussian probability density function with a mean value $\mu_{i,k}$ and a variance $\sigma_{i,k}^2$.

Every new pixel value $z_k$ is checked against the existing $N$ Gaussian distributions to find if a match occurs. A match is defined as a pixel value within the standard deviation (possibly multiplied by a positive constant) of a distribution. When two or more matches occur, only the best matched distribution is chosen by comparing the relative distance from the average value normalized by the standard deviation. If none of the $N$ distributions match the current pixel value, the least probable distribution is replaced with a distribution with the current value as its mean, an initially high variance, and low prior weight given by users.

The weights of the $i$th distributions at time $k$, $\omega_{i,k}$, are adjusted as follows:

$$\omega_{i,k} = (1 - \alpha)\omega_{i,k-1} + \alpha\xi_{i,k} \tag{5}$$

where $\alpha$ is the learning rate, and $\xi_{i,k}$ is 1 for the model that is decided as a best match and 0 for the remaining models. This formulation renormalizes the weights automatically.

The $\mu$ and $\sigma$ parameters for unmatched distributions remain the same. The parameters of the distribution that matches the new observation are updated as follows:

$$\mu_k = (1 - \alpha)\mu_{k-1} + \alpha z_k \tag{6}$$

$$\sigma_k^2 = \min(\sigma_{min}^2, (1 - \alpha)\sigma_{k-1}^2 + \alpha(z_k - \mu_{k-1})^2) \tag{7}$$

where a minimum variance $\sigma_{min}^2$ is introduced as a threshold to keep the variance from decreasing below a minimum value in case there is little change in a pixel

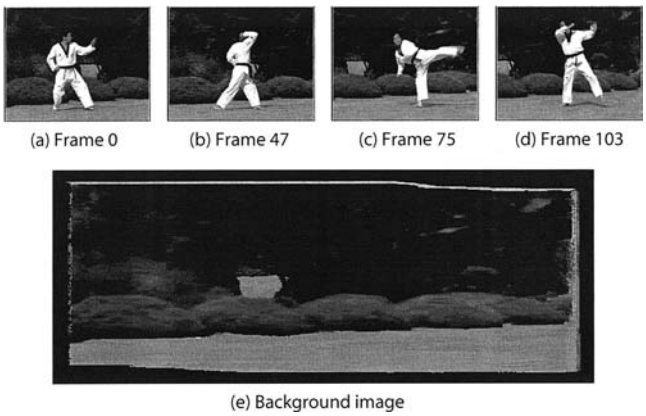


**Figure 4.** *Background image obtained from example input video.*

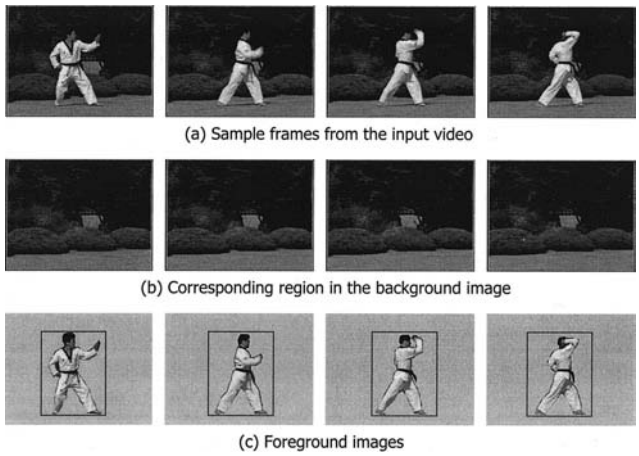


**Figure 5.** *The foreground images from example input video.*

value over a long period of time (Francois & Medioni, 1999).

After all the parameters are updated, we determine the current background distribution as the one with the highest value of $\omega/\sigma$, which means the distribution has the most supporting evidence and the least variance. Thus, the complete background image is obtained when this update procedure is done for all the pixels and for all the frames.[2] Figure 4 shows a background image obtained from an example input video sequence taken from a rotating camera.

## 3 Foreground-Image Generation

The foreground (texture) image for a static foreground object is extracted just once from the background image, by placing a bounding quadrangle around it. To generate a complete foreground image, we first need the boundary information to distinguish the exact portion of the foreground object from the background portion within the foreground image. For effective extraction of the boundary information, we developed a highly interactive image-segmentation tool called *enhanced lane* (Kang & Shin, 2002a). Based on a graph search over the localized window that follows the feature points, the enhanced lane provides both accuracy and time-efficiency in tracking the target boundary interactively, as will be described later in this section. After the boundary extraction, we assign an alpha value of 1 inside the object boundary and 0 elsewhere so that realistic 3D effects can be produced during the navigation.

For a dynamic foreground object, its foreground image is provided at each frame, as the object may change its boundary shape from frame to frame. Thus, foreground-image generation for a dynamic foreground object boils down to boundary extraction at each frame in the video sequence. The boundary information of the dynamic objects can be obtained from the result of the background-detection process. We first warp each frame $k$ using its camera pose information given by the registration process, and generate the difference image $I_k^D$ by computing $|\mathbf{x} - \hat{\mathbf{x}}|$. Then, we can extract the dynamic foreground objects by applying a connected component algorithm (Horn, 1986) with appropriate threshold values for the foreground pixels on this difference image (see Figure 5).

However, this statistical result may be inaccurate (at least partially) in some frames due to noises or ambiguities. Thus, we also provide an iterative boundary-correction scheme, especially for some intermittent se-

[2]After the process, if there are regions in the background still occluded by some foreground pixels (due to insufficient movement), the occluded regions are restored by employing inpainting techniques (Efros & Leung, 1999; Bertalmio, Sapiro, Caselles, & Ballester, 2000).

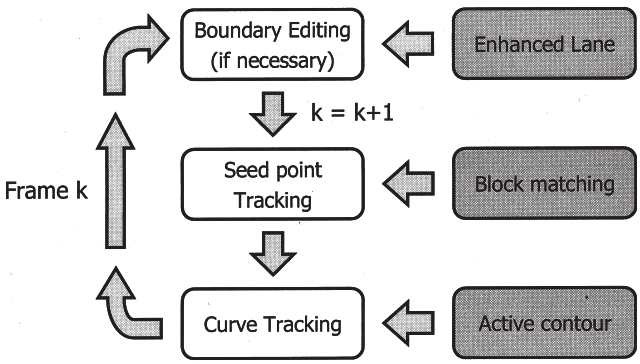


**Figure 6.** *Block diagram for boundary-correction process.*

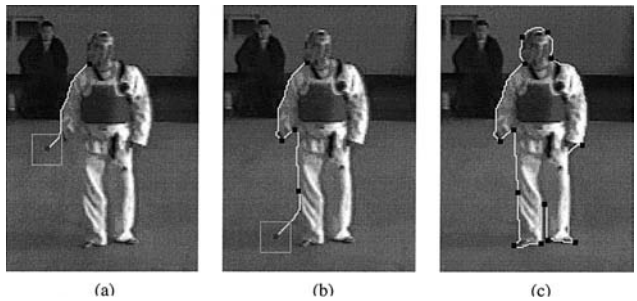


**Figure 7.** *Enhanced lane: (a) The minimum-cost path is displayed as the cursor moves along the boundary; (b) A new seed point is created where the digression is inevitable; (c) The complete boundary is identified.*

quences of frames that have incorrect boundary-segmentation results. As shown in Figure 6, our boundary-correction scheme is composed of three iterative steps. First, the enhanced lane is used to trace the initial boundary curve of the object in the starting frame that needs correction, and the boundary curve is then automatically tracked by snakes in the successive frames. Also, the positions of the seed points in the next frame are estimated by applying block matching, for better guidance of the snake curve (Mitsunaga et al., 1995; Hoch & Litwinowicz, 1996). This automatic boundary-tracking process is repeated in the successive frames until a digression occurs in a certain frame, where the constructed boundary is again postedited by the enhanced lane. After the object boundary is determined for all the frames, the foreground image of the object is constructed by locating a bounding box in which only the foreground pixels inside the boundary are marked as visible (Figure 5c).

### 3.1 Enhanced Lane

Based on a graph search paradigm, the enhanced lane regards an image as a directed graph in which pixel corners and oriented pixel edges represent the vertices of the graph and its arcs, respectively. To each oriented pixel edge, a set of features is assigned to give its local cost. Then, the problem of constructing the best boundary segment between any two points specified on the boundary is reduced to finding the minimum-cost path between the two vertices in the graph. The local cost is computed from the various edge features such as gradient magnitude, gradient direction, Laplacian zero-crossing, etc.

On an input image set as a directed graph, enhanced lane constructs a path map in a local window centered at a seed point on the target boundary. As a cursor moves along the boundary, the minimum-cost path from the seed point to the cursor is dynamically displayed, which gives an impression that the path automatically snaps at the target boundary. The path map is then incrementally updated along the cursor movement to extend the path inside the window sequence, and when a digression occurs the path map is reinitialized with a new seed point to start a new segment. The complete boundary is obtained with a sequence of these path segments comprising a closed path (see Figure 7). Enhanced lane is more powerful than its predecessors such as intelligent scissors (Mortensen and Barrett, 1995) or live wire (Falcao et al., 1998) in that it always guarantees strictly bounded response time regardless of image size, and reduces the digressions by its path map localization, which leads to better accuracy.

Figure 8 shows the strength of our technique. With previous techniques (Mortensen & Barrett, 1995; Falcao et al., 1998), the target path in this figure results in one or more digressions during segmentation since they look only for a globally optimal path. The enhanced lane, however, is more accurate in that it provides a bet-

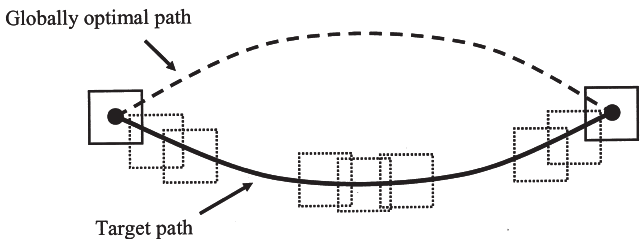


**Figure 8.** *Advantages of enhanced lane.*

ter chance to extract the target path without digressions by localizing the search domain around the target path and performing incremental path-map expansion. Also, the localization of the path map gives better time efficiency than the previous techniques that are based on global graph search. Thus, it is capable of segmenting complex foreground objects from an arbitrary background of a noisy, low-contrasted image with interactive speed. It also provides powerful postediting capability for correcting a part or all of the given boundary curve. For more details on the enhanced lane, see Kang and Shin (2002a).

### 3.2 Block Matching

To estimate the positions of the seed points in the next frame, we adopt a block-matching technique, starting from their positions in the current frame (Tekalp, 1995). Let $I_k$ and $I_{k+1}$ denote the current frame and the next frame, respectively. For each block in $I_k$ centered at a seed point, our goal is to find the most similar block in $I_{k+1}$. In order to find the best match between patterns, we use a variance-normalized correlation as a measure of similarity (Burt, Yen, & Xu, 1982):

$$C_i(\mathbf{x}, \mathbf{d}) = \omega_i(\mathbf{x}, \mathbf{d}) \sum_{\mathbf{n}} F_k(\mathbf{x},\mathbf{n}) F_{k+1}(\mathbf{x} + \mathbf{d}, \mathbf{n}), \quad (8)$$

where

$$F_j(\mathbf{a}, \mathbf{b}) = I_j(\mathbf{a} + \mathbf{b}) - \bar{I}_j(\mathrm{a}) / \sqrt{\sigma_j^2(\mathrm{a})}. \quad (9)$$

Here, correlation $C_i$ is computed between the $i$th block in the current image $I_k$ centered at point x and a pattern in the next image $I_{k+1}$ centered at point $\mathbf{x} + \mathbf{d}$, where n denotes the displacements within the block. $\bar{I}$ stands for the mean value of the block being considered, $\sigma^2$ denotes its variance. Simple multiplication is used as the comparison operator. Thus, finding the best matching block in $I_{k+1}$ is equivalent to computing $\mathbf{d}$, which maximizes $C_i$.

The weight function $\omega_i(\mathbf{x}, \mathbf{d})$ is given as follows:

$$\omega_i(\mathbf{x}, \mathbf{d}) = \beta_i \, |\nabla I_{k+1}^D (\mathbf{x} + \mathbf{d})| + (1 - \beta_i) \max(|\nabla I_{k+1}^D|), \quad (10)$$

where $|\nabla I_{k+1}^D|$ is the gradient magnitude image of $I_{k+1}^D$, that is, the difference image between $I_{k+1}$ and the corresponding region in the background image obtained after the background-detection process as discussed in Section 2. Because this statistically estimated boundary information is not always accurate, we introduce $\beta_i$ ($>0$) to denote its credibility, which is adjusted if the current frame goes through any postediting process:

$$\beta_{i,k+1} = (1 - \alpha)\beta_{i,k} + \alpha\xi_i, \quad (11)$$

where $\alpha$ is the learning rate, and $\xi_i$ is 0 if this point has been modified by the enhanced lane and 1 otherwise. As shown in Equation 10, when the credibility gets low, a fixed value of the maximum gradient is used instead.

### 3.3 Active Contour

The given boundary curve in the current frame is automatically tracked in the next frame by employing active contour or snakes (Kass et al., 1987). A snake is an energy-minimizing curve guided by internal and external forces. The internal force imposes a piecewise smoothness constraint on a snake, and the external force moves the snake toward strong image features such as points, lines, edges, or contours. As image (external) forces for attracting snakes at each frame $k$, we use the gradient map of the difference image $|\nabla I_k^D|$. This can be further convolved with a Gaussian smoothing filter ($G_\sigma * |\nabla I_k^D|$) to attract a distant snake. Also, the positions of the seed points determined by block matching serve as additional external constraints to guide the snake.

Figure 9 shows our iterative boundary-correction process applied to an example frame sequence. The enhanced lane initializes the boundary curve in the first

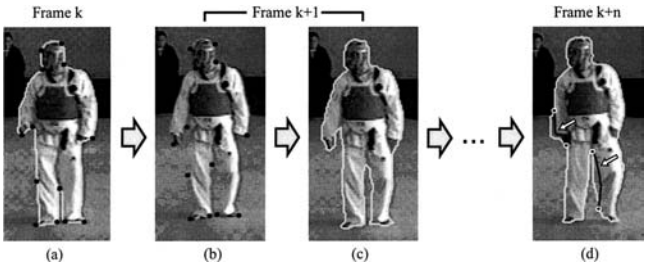


**Figure 9.** *Boundary correction: (a) Enhanced lane; (b) Block matching; (c) Snake; (d) Enhanced lane.*

frame (Figure 9a). In the next frame, the new locations of the seed points (blue dots) are estimated by block matching (Figure 9b), and then the boundary curve is automatically adjusted by snakes (Figure 9c). This automatic boundary-tracking process is repeated from frame to frame until a path digression occurs, which is again postedited by enhanced lane,[3] as indicated by the white arrows in Figure 9d. Note that the interactive postediting process also includes insertion, displacement, and removal of seed points. Figure 10 shows some of the test results of our dynamic boundary-correction method on various sample video sequences.

## 4 Scene-Model Construction

Given the background and the foreground information thus extracted, we construct the 3D scene model consisting of a background model and foreground object models. The background model is obtained from the background image, provided with the vanishing circle. Foreground objects are divided into two types, static or dynamic, each of which is constructed by its own modeling scheme.

### 4.1 Background Model

Since the background image can be thought of as a panoramic (either spherical or cylindrical) image, we

[3]On average, the enhanced lane is applied every ~5–6 frames in our experiments.

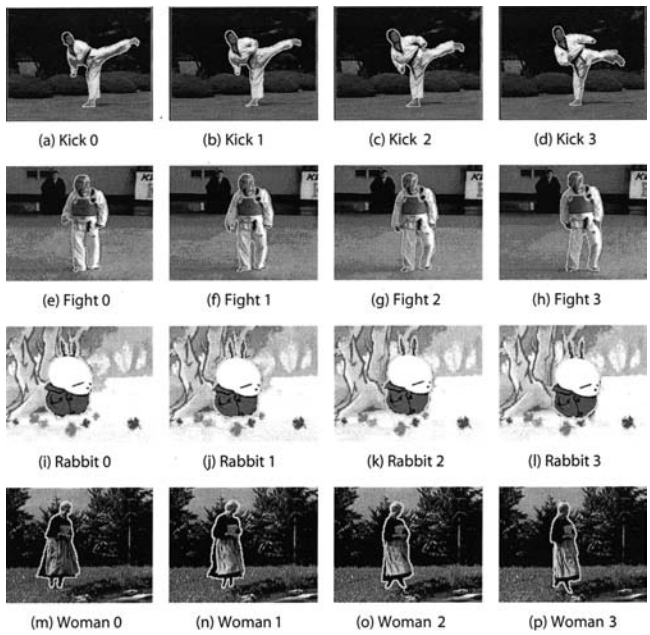


**Figure 10.** *The segmented results for various input video sequences.*

can directly use the modeling scheme for TIPP (Tour Into the Panoramic Picture) (Kang et al., 2001). We first introduce the notion of the vanishing circle, on which our scheme for constructing the background model is based.

**4.1.1 Vanishing Circle.** In obtaining a spherical background image from video, the environment in the video viewed from the camera is mapped onto the base sphere centered at the camera position. Our assumption of an environment with a flat terrain naturally leads to a scene model consisting of a ground plane with the camera (or eye) placed above it. As shown in Figure 11, suppose there are parallel lines $A$ and $B$ on the ground plane. When these lines are viewed from the camera, they are projected onto the base sphere as arcs $A'$ and $B'$, respectively. These arcs $A'$ and $B'$ intersect at a point on the base sphere, which is referred to as a vanishing point. As $A$ and $B$ on the ground plane take on arbitrary inclinations, the set of all vanishing points on the sphere form a circle on the base sphere. This circle is said to be a vanishing circle, and is analogous to the

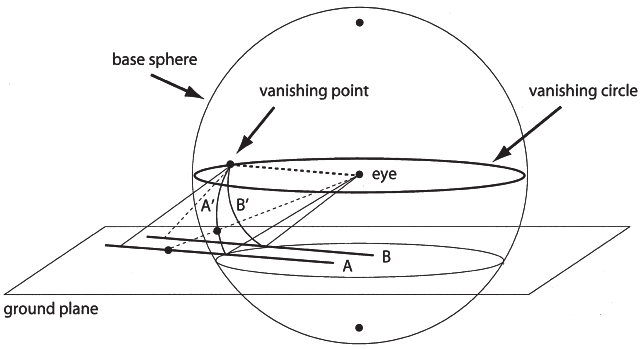


**Figure 11.** *Vanishing point and vanishing circle.*

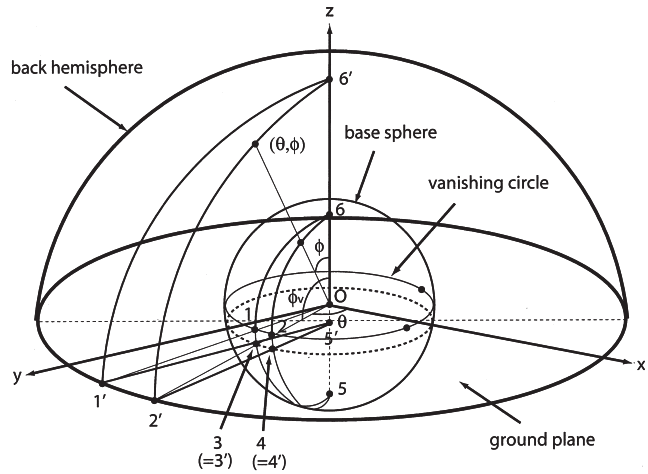


**Figure 12.** *Point correspondences between the base sphere and the background model.*

vanishing line for a planar image. The vanishing circle for a cylindrical base object can be obtained in a similar way.

**4.1.2 Model Construction.** The vanishing circle divides the base sphere into two disjoint hemispheres. The lower hemisphere corresponds to the ground plane in the 3D environment, and the upper hemisphere corresponds to the space above the ground plane. Thus, the vanishing circle can be thought of as the horizon that separates the earth, represented by the ground plane, from the sky. If we inversely project the vanishing circle back to the scene, it is mapped to a set of points at infinity on the ground plane. Each of these points is an ideal point in the direction from the viewpoint to a point on the vanishing circle.

Based on this observation, we first specify the location of the vanishing circle and then project the upper hemisphere of the base sphere on the *back hemisphere,* which has an arbitrarily large radius centered at the camera position. The lower hemisphere is projected onto the ground plane. In practice, the back hemisphere is set to have some finite radius to avoid computational difficulty. This is equivalent to slightly moving down the vanishing circle to set the intersection between the ground plane and the hemisphere at a finite distance.

The correspondence between the points on the base sphere (~1–6) and those on the background model (~1–6′) is shown in Figure 12. We assume that the base sphere and the camera are centered at the origin, the initial camera view-up vector is toward the $+z$ direction, the ground plane is parallel to the $x$-$y$ plane, and the height of the camera from the ground plane is $h$. Thus, a point $(\theta, \phi)$ on a base sphere is projected to the point $(-h \cos\theta \sin\phi/\cos\phi_v, -h \sin\theta \sin\phi/\cos\phi_v, -h \cos\phi/\cos\phi_v)$ on the back hemisphere if $\phi < \phi_v$; otherwise, it goes to $(h \cos\theta \tan\phi, h \sin\theta \tan\phi, -h)$ on the ground plane, where $\phi_v$ denotes the angle between the positive $z$-axis and the vector from the origin to a point on the vanishing circle (Kang et al., 2001). Similarly, Figure 13 shows the correspondence between the points on the base cylinder and those on its background model. In this case, the vanishing circle divides the base cylinder into lower and upper cylinder, which correspond to the ground and the space above, respectively. Note that with a cylindrical base object, holes appear at the centers of the two disjoint regions.

Based on the above correspondence formulation, we can now map each pixel $(\theta, \phi)$ on the background image onto the corresponding point on the 3D background model. Figure 14 shows a background model constructed from an example video sequence. We intentionally used hand-drawn video frames to clearly show the relationships between the individual frames and the

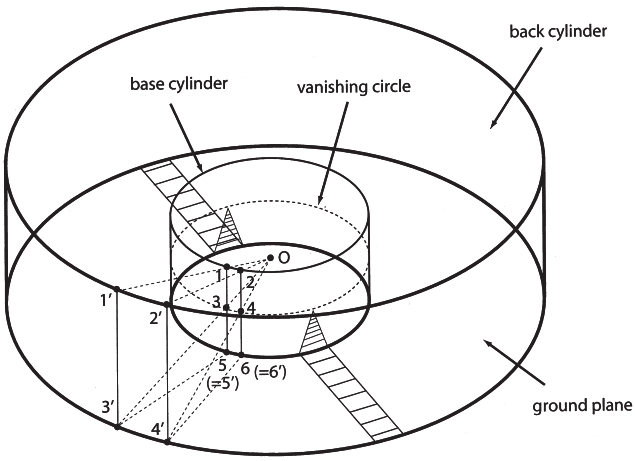


**Figure 13.** *Point correspondences between the base cylinder and the background model.*

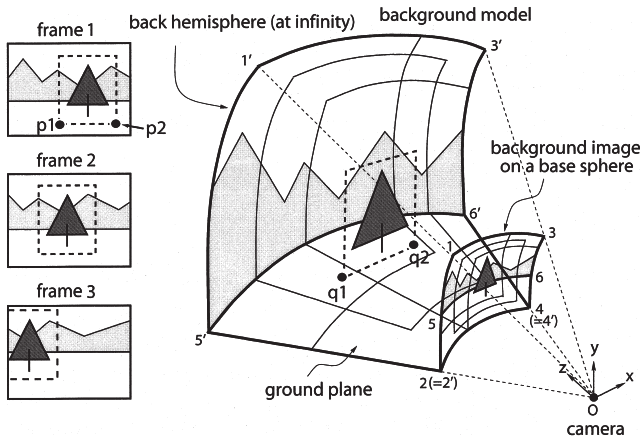


**Figure 15.** *Static foreground model for a video sequence.*

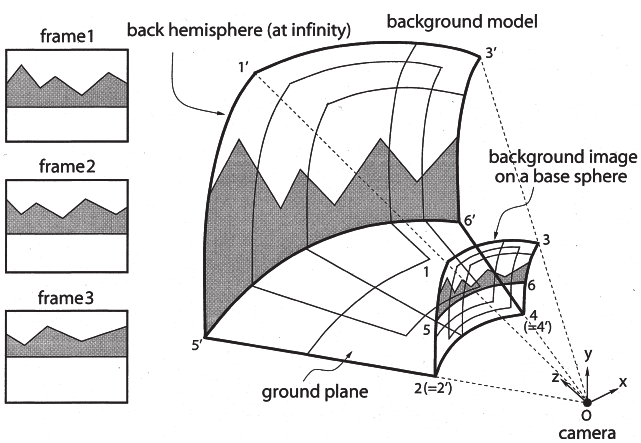


**Figure 14.** *Background model constructed from a video sequence.*

background model. Note that only a part of the base sphere (and thus the background model) is covered by the given frames. The actual mapping from the video frames to the background model is implemented by texture mapping. The background texture image is obtained by first segmenting out the static foreground objects from the background image and filling in the holes by inpainting techniques (Efros & Leung, 1999; Bertalmio et al., 2000). We divide the image into two disjoint subimages using the line on the background image corresponding to the vanishing circle of the base sphere. The upper subimage serves as a texture map for the back hemisphere, and the lower subimage for the ground plane.

When we use hardware texture mapping, we have to make sure all the texture maps are for linear (line-to-line) perspective mapping. Since the spherical image basically contains a line-to-arc map, we cannot directly use the lower subimage as a texture map for the ground plane. Thus, we first need to convert the lower subimage into a linear map, using a projective mapping from the base sphere onto the ground plane. However, the upper subimage does not need this type of preprocessing because it will be used for sphere-to-sphere mapping, which is essentially a linear mapping because the back hemisphere is implemented as a tessellated polyhedron for hardware texture mapping.

## 4.2 Foreground Model

**4.2.1 Static Foreground Model.** A static foreground object extracted from the background image is modeled as a polygon (usually a quadrangle bounding the extracted object) standing on the ground plane (see Figure 15). We first compute the 3D coordinates for the two vertices of the polygon on the ground plane, which form the bottom edge (points $\mathbf{q}_1$ and $\mathbf{q}_2$ in the figure). These points have corresponding points $\mathbf{p}_1$ and

$\mathbf{p}_2$ on an original frame, which are projected onto the base sphere using the camera-pose information obtained from the image-registration process (as described in Section 2.2). Then the 3D coordinate for $\mathbf{q}_1$ is computed by the simple intersection test between the ground plane and the line connecting O (camera) and the point on the base sphere projected from $\mathbf{p}_1$. The coordinate for $\mathbf{q}_2$ is obtained similarly. The 3D coordinates of these bottom vertices automatically give the depth information of the foreground polygon. Note that these two vertices on the ground may be assigned different depth values, which means the foreground polygon is not necessarily parallel to the view plane. Assuming that the foreground polygon stands vertically to the ground plane, the coordinates of the remaining vertices in the polygon are also automatically computed using similar intersection tests.

As in the case of the background model, the rendering of the foreground objects is performed by texture mapping. The foreground (texture) image for each object corresponds to the region inside the foreground polygon in the background image. Since the spherical background image is a nonlinear texture map, we convert it into a linear map to correct the foreground image. To do this, we first set the bounding rectangle of a foreground polygon on its corresponding 3D plane in the background model. Then we project onto this rectangle the portion of the background image that is visible from the camera through the rectangle. The resulting image on the rectangle provides a correct linear map to be used as a foreground texture image. As described in Section 3, we assign alpha values of 1 inside the segmented portion of the foreground texture and 0 elsewhere, to show clear 3D effects around the object boundaries during scene navigation.

As proposed in Horry et al. (1997), a foreground object can have a hierarchical structure in a more complex environment (Figure 16a). Also, if a foreground object has a curved structure at the lower boundary on the ground plane, it is approximated by a group of piecewise planar models. This type of model is especially useful for an object that spans a wide range in a horizontal direction (Figure 16b). When there are foreground objects occluding others in the image, the occluded portions of the foreground objects should be restored to generate an image from a new viewpoint, by employing inpainting techniques (Efros & Leung, 1999; Bertalmio et al., 2000). Thus, multiple foreground images are required in this case so that each occluded object can be associated with its corresponding foreground image (Figure 16c).

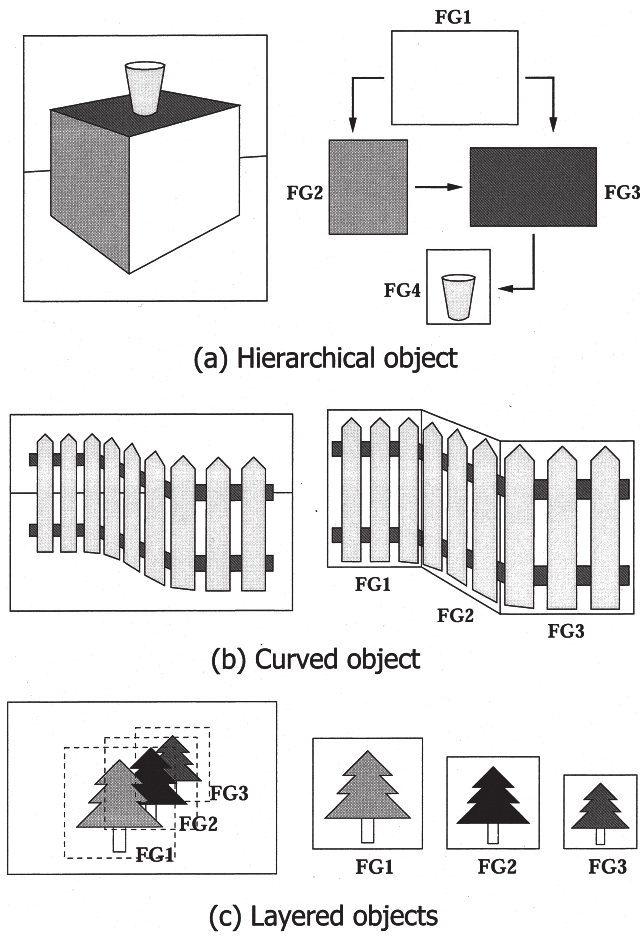


**Figure 16.** *Extended foreground models.*

**4.2.2 Dynamic Foreground Model.** Dynamic foreground objects are also modeled as polygons and attached to the constructed background model. Because they do not appear in the background image and they change their shapes and locations from frame to frame, an independent foreground model should be con-

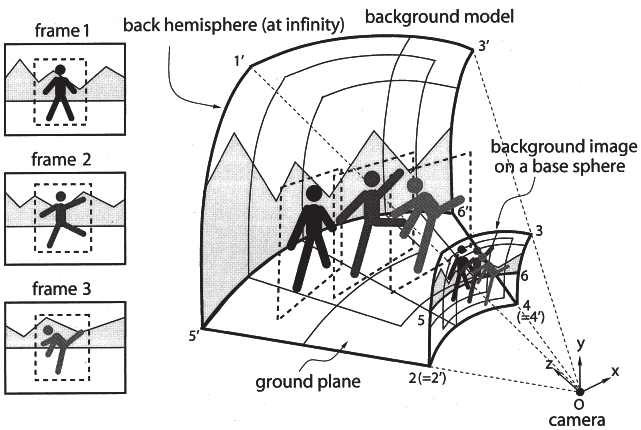


**Figure 17.** *Dynamic foreground models for a video sequence.*


**Figure 18.** *Tour into the video with a fixed camera.*

structed at each frame. The foreground polygons in each frame are first mapped onto the the base object using the transformations obtained by the registration process. Then their vertex coordinates in the scene model are computed similarly, as described in Section 4.2.1. After all the frames are processed, we obtain a sequence of foreground polygons that have independent shapes and locations for each frame. During the rendering process for an output video production, all the dynamic foreground objects are played back in the scene, that is, they are placed on the scene model and rendered one frame at a time, as shown in Figure 17. Note that we once again assign an alpha value of 1 only inside the segmented portion of the dynamic foreground texture at each frame, to show clear boundaries for the objects during rendering.

## 5 Experimental Results

Figure 18 shows the result that our TIV scheme produced with a sample video. The input video contains dynamic motion of foreground objects with a fixed camera. Figures 18a through 18d show the initial frame, the background image, the specifications for the vanishing circle and the foreground polygons, and the segmented results of the foreground objects. The background model is constructed by specifying the vanishing circle in the background image. The boundaries of foreground objects are extracted for each frame as described in Section 3. Once the scene model is constructed, the user can interactively navigate the scene by controlling the camera position and orientation. Figures 18e through 18h show samples from the original video sequence, and Figures 18i through 18l show the corresponding walk-through images seen from another viewpoint. Note that 3D effects are achieved from camera navigation due to the depth information assigned to foreground objects and the difference in viewpoints.

Figure 19 illustrates another result of TIV with a sample video containing a camera motion. Figures 19a through 19d show some of the initial frames. The background image is first generated after frame-by-frame image registration and automatic background detection, and the scene model is then constructed given the background image. Figures 19e through 19h show walk-through images seen from another viewpoint at each corresponding frame. Figures 19i through 19l show an example synthetic video sequence obtained by inserting a virtual dynamic object, and Figures 19m through 19p show another synthetic video sequence after inserting another virtual dynamic object imported from other video source.

The input video in Figure 20 is a cartoon animation clip. Cartoon animation can be thought of as the most

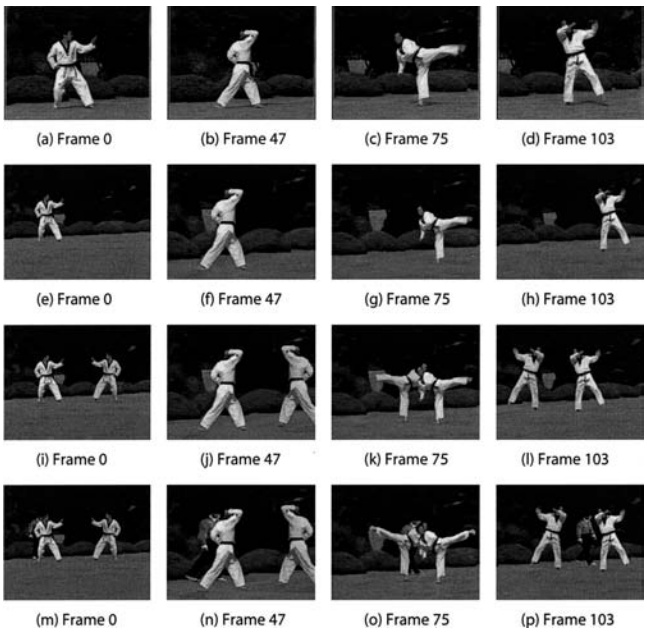


**Figure 19.** *Tour into the video with a moving camera.*

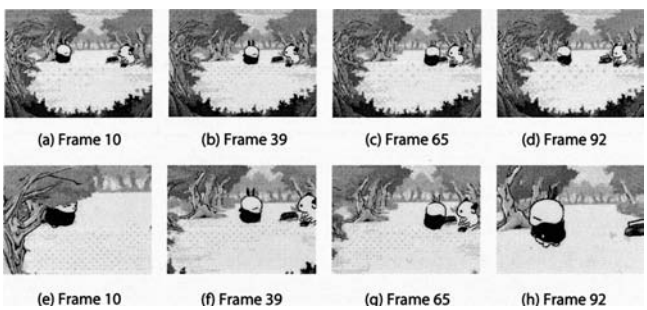


**Figure 20.** *Tour into the cartoon animation.*

suitable input for TIV because it usually consists of a set of conspicuous 2D dynamic objects with a relatively static background. Thus, we can achieve both efficiency and robustness in going through each of the three processes, background detection, scene-model construction, and boundary segmentation. In Figure 21, TIV is applied to a video clip where the background is also dynamic (because of the flow of water). To model the dynamic background, we use a video texture rather than a static texture, that is, a minimum number of frames are extracted and used to cover the periodic movement of

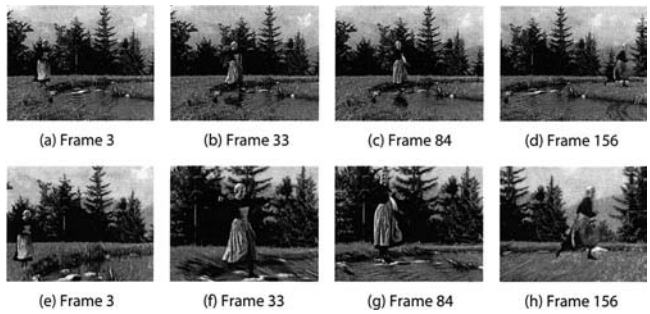


**Figure 21.** *Tour into the video with dynamic background.*

the water.[4] Thus, the output video sequence can have all three types of movements simultaneously, including those of the foreground, the background, and the camera.

The rendering speed is dependent on the number of foreground objects in the image and the image size. The entire scheme is implemented in C++ with OpenGL library on Intel Pentium PC (PIII 800 MHz processor and 512 MB memory) equipped with nVIDIA GeForce2GTS graphics processor. On average, the output sequence of images with 640×480 pixels is generated at an interactive rate (over 100 frames/s). Note that the scene navigation is done in real time once the complete dynamic scene model is constructed as a preprocess. In our experiments, the preprocessing time for each input video took less than an hour, although it could vary depending on the length of the video and the number of objects in the scene. In general, the semiautomatic boundary correction (described in Chapter 3) takes up most of the preprocessing time, especially when the input video contains objects with unclear boundaries.

## 6 Conclusions and Future Work

We have proposed a novel scheme for producing a sequence of walk-through images from a video

[4]Although we used the video-texture-based dynamic background only in this example, it could also be used in other input videos, especially to recover shadows.

stream of a dynamic scene. To generalize the original idea of TIP to video input, our scheme is designed to have the following three components: background detection, foreground extraction, and scene model construction. Assuming that the input video contains no strong parallax effects, we apply an automatic background-detection algorithm based on a 4-parameter motion model that deals with camera rotation and zoom. While the foreground objects are also extracted from the video as a by-product of the background-detection process, we have also provided an efficient, iterative boundary-correction mechanism based on active contour and enhanced lane. Finally, we have incorporated a new 3D model-construction method based on a vanishing circle detected in the scene. Our scheme also facilitates an augmented video production by allowing static or dynamic foreign objects to be imported into the scene.

The general objective of our scheme is to let users produce, through interactive control, walk-through images from a given video sequence in real time. Thus, in a sense, it enables people to experience the feel of 3D navigation in an originally 2D video. Another goal is to help them synthesize a new augmented video by incorportating virtual objects in the original dynamic scene. While our scheme employs various automatic techniques to minimize most of the tedious jobs for handling an excessive number of frames in the sequence, it is also designed to provide users with highly interactive control so that diverse results can be produced from a single input video, reflecting their own interpretation and imagination. The proposed scheme can process various types of videos containing dynamic scenes, such as sports coverage, cartoon animation, and movie films, in which objects are changing their shapes and locations continuously in a relatively static background. The possible applications of our scheme include vision-based Virtual Reality (VR), augmented video production, advanced video editing, virtual tour systems, etc.

We can think of a number of research topics for further extensions. In our current implementation, we made the simplifying assumptions that the input video contains a negligible amount of motion parallax effects and that dynamic objects move on a flat terrain. In order to deal with a video with significant parallax effects, our motion model should be extended to incorporate camera displacement also. Thus, the resulting mosaic representation would include the intensity image plus a corresponding depth map for a scene (Irani, Anandan, & Hsu, 1995). This type of representation is also useful for constructing more sophisticated foreground models (Shade, Gortler, He, & Szeliski, 1998; Oh, Chen, Dorsey, & Durand, 2001). For dynamic objects that are not moving on the ground (e.g., jumping or flying objects), additional information should be provided (either automatically or interactively) to remove ambiguity in computing their 3D locations. For example, the depth of a jumping object could be inferred from neighboring frames in which the object touches the ground or other objects.

As shown in some experimental results, the background image in the scene model can also be made dynamic by employing video texture (Schodl, Szeliski, Salesin, & Essa, 2000), that is, a texture map consisting of a sequence of time-coherent images. This is especially useful for describing a scene with a dynamic background, restoring shadows, and creating an infinite stream of images with either random play or video loops. Finally, when a part of the scene is occluded by a foreground object, the occluded portion (or "hole") could be restored by employing automatic hole-filling techniques such as texture synthesis and image inpainting (Efros & Leung, 1999; Bertalmio et al., 2000). Texture-synthesis techniques could also be used to extrapolate the information outside the background region covered by the footage, which may provide more immersive effects during camera navigation (Efros & Leung, 1999).

## Acknowledgments

For the first author, this research was supported by University IT Research Center Project, and for the second author by the National Research Laboratory Program of the Ministry of Science & Technology, and the Brain Korea 21 Program of the Ministry of Education & Human Resources Development.

## References


Azuma, R., 1997. A survey of augmented reality. *Presence: Teleoperators and Virtual Environments, 6*(4), 355–385.

Azuma, R., Baillot, Y., Behringer, R., Feiner, S., Julier, S., & MacIntyre, B., 2001. Recent advances in augmented reality. *IEEE Computer Graphics and Applications, 21*(6), 34–47.

Beardsley, P., Torr, P., & Zisserman, A., 1996. 3D model aquisition from extended image sequences. *Proceedings of the 4th European Conference on Computer Vision,* 683–695.

Bergen, J., Anandan, P., Hanna, K., & Hingorani, R., 1992. Hierarchical model-based motion estimation. *Proceedings of the 2nd European Conference on Computer Vision (ECCV '92),* 237–252.

Bertalmio, M., Sapiro, G., Caselles, V., & Ballester, C., 2000. Image inpainting. *ACM SIGGRAPH 2000 Conference Proceedings,* 417–424.

Burt, P., Yen, C., & Xu, X., 1982. Local correlation measures for motion analysis: A comparative study. *IEEE Conference on Pattern Recognition and Image Processing,* 14–17.

Chen, S., 1995. QuickTime VR: An image-based approach to virtual environment navigation. *ACM SIGGRAPH '95 Conference Proceedings,* 29–38.

Chen, S., & Williams, L., 1993. View interpolation for image synthesis. *ACM SIGGRAPH '93 Conference Proceedings,* 279–288.

Darsa, L., Silva, B., & Varshney, A., 1995. Navigating static environments using image-space simplification and morphing. *Proceedings of the 1997 Symposium on Interactive 3D Graphics,* 25–34.

Efros, A., & Leung, T., 1999. Texture synthesis by non-parametric sampling. *Proceedings of the IEEE International Conference on Computer Vision,* 1033–1038.

Falcao, A., Udupa, J., Samarasekera, S., Sharma, S., Hirsch, B., & Lotufo, R., 1998. User-steered image segmentation paradigms: Live wire and live lane. *Graphical Models and Image Processing, 60*(4), 233–260.

Fitzgibbon, A., & Zisserman, A., 1998. Automatic 3D model aquisition and generation of new images from video sequences. *Proceedings of the European Signal Processing Conference (EUSIPCO '98),* 1261–1269.

Francois, A., & Medioni G., 1999. Adaptive color background modeling for real-time segmentation of video streams. *Proceedings of the International Conference on Imaging Science, Systems, and Technology,* 227–232.

Friedman, N., & Russell, S., 1997. Image segmentation in video sequences: A probabilistic approach. *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence.*

Gortler, S., Grzeszczuk, R., Szeliski, R., & Cohen, M., 1996. The lumigraph. *ACM SIGGRAPH '96 Conference Proceedings,* 43–54.

Hartley, R., & Zisserman, A., 2000. *Multiple view geometry in computer vision.* Cambridge, UK: Cambridge University Press.

Hoch, M., & Litwinowicz, P., 1996. A semi-automatic system for edge tracking with snakes. *Visual Computer, 12*(2), 75–83.

Horn, B., 1986. *Robot vision.* Cambridge, MA: MIT Press.

Horry, Y., Anjyo, K., & Arai, K., 1997. Tour into the picture: Using a spidery mesh interface to make animation from a single image. *ACM SIGGRAPH '97 Conference Proceedings,* 225–232.

Irani, M., Anandan, P., & Hsu, S., 1995. Mosaic based representations of video sequences and their applications. *Proceedings of ICCV '95,* 605–611.

Kang, H., Pyo, S., Anjyo, K., & Shin, S., 2001. Tour into the picture using a vanishing line and its extension to panoramic images. *EuroGraphics 2001 Conference Proceedings,* 132–141.

Kang, H., & Shin, S., 2002a. Enhanced lane: Interactive image segmentation by incremental path map construction. *Graphical Models, 64*(5), 282–303.

Kang, H., & Shin, S., 2002b. Tour into the video: Image-based navigation scheme for video sequences of dynamic scenes. *ACM VRST 2002 Conference Proceedings,* 73–80.

Kass, M., Witkin, A., & Terzopoulos, D., 1987. Snakes: Active contour models. *Proceedings of the First International Conference on Computer Vision,* 259–268.

Koller, D., Weber, J., Huang, T., Malik, J., Ogasawara, J., Rao, B., & Russell, S., 1994. Towards robust automatic traffic scene analysis in real-time. *Proceedings of the International Conference on Pattern Recognition,* 126–131.

Levoy, M., & Hanrahan, P., 1996. Light field rendering. *ACM SIGGRAPH '96 Conference Proceedings,* 31–42.

Liebowitz, D., Criminisi, A., & Zisserman, A., 1999. Creating architectural models from images. *EuroGraphics '99 Conference Proceedings,* 39–50.

Lippman, A., 1980. Movie maps: An application of the optical videodisc to computer graphics. *ACM SIGGRAPH '80 Conference Proceedings,* 32–43.

McMillan, L., & Bishop, G., 1995. Plenoptic modeling: An image-based rendering system. *ACM SIGGRAPH '95 Conference Proceedings,* 32–43.

Milgram, P., Shumin, S., Drascic, D., & Grodski, J., 1993.

Applications of augmented reality for human-robot communication. *Proceedings of the International Conference on Intelligent Robots and Systems,* 1467–1472.

Miller, G., Hoffert, E., Chen, S., Patterson, E., Blackketter, D., Rubin, S., Applin, S., Yim, D., & Hanan, J., 1992. The virtual museum: Interactive 3D navigation of a multimedia database. *Journal of Visualization and Computer Animation, 3,* 183–197.

Mitsunaga, T., Yokoyama, T., & Totsuka, T., 1995. AutoKey: Human assisted key extraction. *ACM SIGGRAPH '95 Conference Proceedings,* 265–272.

Mortensen, E., & Barrett, W., 1995. Intelligent scissors for image composition. *ACM SIGGRAPH '95 Conference Proceedings,* 191–198.

Oh, B., Chen, M., Dorsey, J., & Durand, F., 2001. Image-based modeling and photo editing. *ACM SIGGRAPH 2001 Conference Proceedings,* 433–442.

Press, W., Flannery, B., Teukolsky, S., & Vetterling, W., 1992. *Numerical recipes in C: The art of scientific computing.* Cambridge, UK: Cambridge University Press.

Schodl, A., Szeliski, R., Salesin, D., & Essa, I., 2000. Video textures. *ACM SIGGRAPH 2000 Conference Proceedings,* 489–498.

Shade, J., Gortler, S., He, L., & Szeliski, R., 1998. Layered depth images. *ACM SIGGRAPH '98 Conference Proceedings,* 231–242.

Shum, H., & Szeliski, R., 1998. Construction and refinement of panoramic mosaics with global and local alignment. *Proceedings of the Sixth International Conference on Computer Vision (ICCV'98),* 953–958.

Sloan, P., Cohen, M., & Gortler, S., 1997. Time critical lumigraph rendering. *Proceedings of the Symposium on Interactive 3D Graphics,* 17–23.

Stauffer, C., & Grimson, W., 1999. Adaptive background mixture models for real-time tracking. *Proceedings of the IEEE Computer Vision and Pattern Recognition,* 246–252.

Tekalp, A., 1995. *Digital video processing.* Upper Saddle River, NJ: Prentice Hall.

Tomasi, C., & Kanade, T., 1992. Shape and motion from image streams under orthography: A factorization method. *International Journal of Computer Vision,* 137–154.

Vieren, C., Cabestaing, F., & Postaire, J., 1995. Catching moving objects with snakes for motion tracking. *Pattern Recognition Letters, 16,* 679–685.

Wren, C., Azarbayejani, A., Darrell, T., & Pentland, A., 1997. Pfinder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 19*(7), 780–785.