EUROGRAPHICS 2001 / A. Chalmers and T.-M. Rhyne
(Guest Editors)

Volume 20 (2001), Number 3

Tour Into the Picture using a Vanishing Line and its Extension to Panoramic Images

Hyung Woo Kang[†], Soon Hyoung Pyo[‡], Ken-ichi Anjyo[§], and Sung Yong Shin[†]

[†]Department of Computer Science, Korea Advanced Institute of Science and Technology
[‡]Electronics and Telecommunications Research Institute
[§]OLM Digital, Inc.
e-mail: [†]*{henry, syshin}@jupiter.kaist.ac.kr*, [‡]*shpyo@etri.re.kr*, [§]*anjyo@acm.org*

Abstract

Tour into the picture (TIP) proposed by Horry et al.[13] *is a method for generating a sequence of walk-through images from a single reference picture (or image). By navigating a 3D scene model constructed from the picture, TIP produces convincing 3D effects. Assuming that the picture has one vanishing point, they proposed the scene modeling scheme called spidery mesh. However, this scheme has to go through major modification when the picture contains multiple vanishing points or does not have any well-defined vanishing point. Moreover, the spidery mesh is hard to generalize for other types of images such as panoramic images. In this paper, we propose a new scheme for TIP which is based on a single vanishing line instead of a vanishing point. Based on projective geometry, our scheme is simple and yet general enough to address the problems faced with the previous method. We also show that our scheme can be naturally extended to a panoramic image.*

Keywords: *Image-based modeling/rendering, projective geometry, vanishing line, panoramic image.*

1. Introduction

1.1. Motivation

Real-time generation of photorealistic images is a recurring theme in computer graphics. Several rendering techniques, such as ray-tracing or radiosity, have been developed to create images of high quality. However, these techniques may not be applicable to real-time rendering for a scene of complex geometry, since their image generation time depends on the geometric complexity of the scene. Moreover, it takes excessive human effort to model the complex scene itself.

Recently, a different approach for real-time realistic image generation, called image-based rendering, has been of much attention. In this approach, an image viewed from an arbitrary viewpoint is synthesized using scene information extracted from a set of reference images. Compared to traditional rendering, this approach has several advantages: It does not require modeling of complex geometry of a scene. It provides real-time scene rendering since the image generation time depends only on the size of the image. Moreover, a high-quality image can be generated when photorealistic images are used as reference images.

Tour into the picture (TIP) proposed by Horry et al.[13] is a method that facilitates image-based navigation into a single reference picture (or image). This method constructs a simple 3D scene model from a 2D image and generates realistic walk-through images by navigating the model. They proposed an interactive modeling scheme called spidery mesh based on the location of a vanishing point in the image. However, since their scheme assumes that the image has one vanishing point, major modification is required when the image contains multiple vanishing points or no clearly-identified vanishing point. Moreover, the spidery mesh is hard to generalize for other types of images such as panoramic images.

In this paper, we present a new modeling scheme for TIP based on a single vanishing line rather than a single vanishing point. A vanishing line contains, according to projective geometry [16], all vanishing points caused by parallel lines in a plane. The scene model is constructed with respect to the vanishing line interactively specified by the user. The new scheme is simpler than the spidery mesh, and yet general enough to deal with the problems faced with the previous method. Also, we show that our scheme can be naturally extended to navigation into a panoramic image.

1.2. Related work

McMillan et al. explained image-based rendering with the notion of a plenoptic function, which defines the radiant energy at each point in the space through every incident direction [7]. In this point of view, reference images are samples of the plenoptic function. An image-based rendering technique reconstructs the plenoptic function by interpolating those samples. An image viewed from an arbitrary viewpoint is generated by resampling the plenoptic function thus obtained.

An early approach to image-based rendering is to employ environment maps as samples of the plenoptic function at selected locations [1, 2, 3, 4]. To navigate the environment, one has to move the viewpoint discretely from location to location while continuously changing the viewing direction. That is, continuous camera translation is not supported in this approach. To avoid this problem, a warp function is employed to establish the relative movement of each pixel with respect to camera movement [5, 6, 7, 8]. In general, a reliable warp function is hard to obtain automatically for a real environment whose exact geometry is not available. A different approach is to construct a light field from a set of plenoptic samples such as multiple reference images taken at regular grid points [9, 10, 11]. This approach requires an efficient way to maintain dense plenoptic samples.

Debevec proposed an efficient method for modeling existing architectural scenes from a sparse set of still photographs [12]. A basic geometric model of the scene is built with primitives such as boxes, prisms, and surfaces of revolution, and then their dimensions are optimized to match the input photographs. Additional geometric detail of the model is automatically recovered through stereo matching. They used a view-dependent texture mapping method to render a novel view of the scene from the desired location.

When the input image is full of man-made structures (e.g. buildings and houses) a rather complex 3D model can be constructed from even a single image. Liebowitz et al. presented algorithms useful for building architectural models from a single image [15]. Using various geometric constraints that are common especially in architectural scenes such as parallelism and orthogonality, these algorithms are able to measure plane orientations, point locations, internal camera parameters, etc. Based on measured data, the architectural scene can be recovered from a single image. This approach rather focuses on accurately reconstructing the foreground object from a single image of architectural scene with artificial regularities, and is not intended for image-based navigation for non-sophisticated users.

1.3. Overview

Given a single reference image, our goal is to provide a simple, efficient, and general modeling scheme and an interface that gives convincing visual effects of traveling into the scene, rather than to accurately reconstruct the original structure of objects from the image. While the spidery mesh of TIP is usually suited for an image with a single vanishing point, our modeling scheme is directly applicable to a larger set of images which contain clear or unclear vanishing points. Moreover, our modeling scheme is naturally extended to navigation into a panoramic image.

The scene model consists of a background model and foreground object models. The background model is constructed based on the location of a vanishing line, and the foreground object models are then placed on the ground plane of the background model, exploiting their relationship to the camera position and the ground plane. By interactively specifying the geometric constraints in the scene through simple interface, users can make their imagination real as they find or feel in the 2D image. We propose simple methods to compute all the 3D coordinates for the models, and to generate their corresponding texture maps (called background image and foreground images) from the input image. The idea for planar images nicely carries over to panoramic images by defining “vanishing circles” that play the same role of vanishing lines.

The remainder of this paper is organized as follows. In Section 2, we briefly discuss the method of Horry et al. Section 3 presents how our background model is constructed based on the notion of a vanishing line. Section 4 deals with the foreground models that are to be attached to the background model to form a scene. We discuss how to generate walkthrough images from the constructed scene model in Section 5. In Section 6, we show how to adapt our scheme to a panoramic image. Section 7 shows the experimental results both for planar and panoramic images. Finally, we conclude this paper in Section 8.

2. Tour into the picture

TIP proposed by Horry et al. is a method to create an image viewed from a camera which can be walked-through or flown-through a simple 3D scene model constructed from an input image. The overall flow diagram for this method is illustrated in Fig. 1. To generate an output image viewed from a new viewpoint, we must construct a 3D scene model to be rendered, and obtain the images containing the texture information of the model.

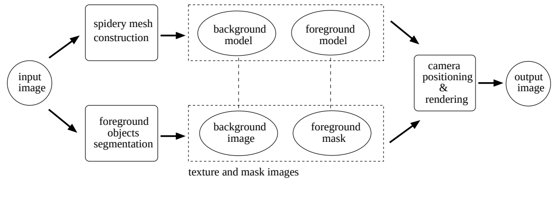

Figure 1: *Flow diagram for the original TIP*

The scene model is composed of a background model together with the foreground object models. To interactively construct the scene model, they proposed a modeling scheme called spidery mesh. In a basic form of spidery mesh, the location of the vanishing point as well as that of the inner rectangle are interactively specified (Fig. 2a), and the image is decomposed into five polygons based on the specification (Fig. 2b). The background model is then constructed with five rectangles in the 3D space which correspond to these five polygons (Fig. 2c). These rectangles represent the floor, the ceiling, the left wall, the right wall, and the rear wall, respectively. The authors also suggested a simplified version of spidery mesh for an image which has no clearly identified vanishing point. In this case, the background model is constructed with just the floor and the rear wall. Later, they added another type of the spidery mesh that can be applied to images with two vanishing points [14]. Those types of spidery mesh have quite different structures specialized for corresponding input images. After the background model is constructed, the foreground objects are also interactively identified in the image. They are modeled as polygons in the 3D space and attached to the background model by computing their 3D coordinates in the scene model.

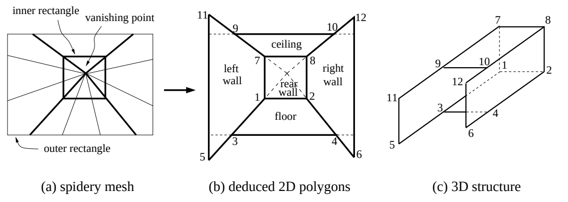

Figure 2: *Spidery mesh and the background model of the original TIP*

Together with the scene model, the images called background image and foreground mask are generated by segmenting the foreground objects from the input image. The background image is used as a texture which is to be mapped onto the background model. This image is obtained by erasing the region occupied by the foreground objects and painting the region with neighboring colors, using conventional 2D painting tools. For foreground objects, the input image is used as their texture map. However, additional information is needed to distinguish the exact portion of the foreground object from the background portion within the polygonal foreground model. This information is recorded in a gray-scale image called foreground mask, where it is white in the region for the foreground object and black elsewhere. By properly assigning the alpha value of the white colored pixels in the mask, we can control the opacity of the foreground object. Thus, an image viewed from a novel viewpoint is created by choosing the camera position and rendering the constructed scene model. The color of each pixel in the output image is decided by referring to the background image, the input image, and the foreground mask.

3. Background model

3.1. Vanishing line

A vanishing point is formed in an image by a set of parallel lines. In Fig. 3, the real objects lie on a plane called object plane (e.g., the horizontal plane on which the viewer stands). A point a in the object plane appears as a point a' in the image plane which is an intersection point between the image plane and the line connecting a and the eye point. In the similar way, the line A on the object plane appears in the image plane as an intersection line A' between the image plane and the plane π_A which contains A and the eye position.

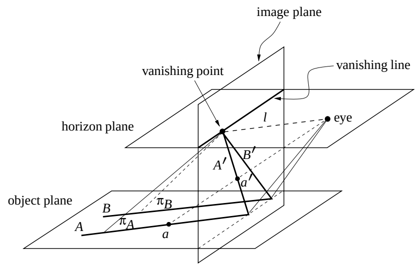

Figure 3: *vanishing point and vanishing line*

As shown in the figure, lines A and B on the object plane are parallel, and appear in the image as A' and B', respectively. Since the two planes π_A and π_B (the one containing A and the eye point and the other containing B and the eye point) are not parallel, A' and B' intersect at a point in the image plane called a *vanishing point*. In fact, the vanishing point is the point at which the intersection line l of π_A with π_B meets the image plane. Since A and B are parallel on the object plane, l is parallel to A, B and the object plane. Hence, l lies on the plane which contains the eye point and is parallel to the object plane. This plane is called a horizon plane. As the lines A and B in the object plane take on all possible inclinations (relative to the image plane), l also takes on the same inclinations on the horizon plane as those of A and B, describing a line of vanishing points in the image plane. This line is called a *vanishing line*. It is the intersection of the image plane with the horizon plane. For example, the horizon that we can see through the landscape, is a vanishing line formed by all parallel lines on the object plane.

We can obtain important information from the location of the vanishing line in an image. Since the horizon plane which contains the vanishing line and the eye point is always parallel to the object plane, the line starting from the eye position and intersecting the image plane below the vanishing line always meets the object plane. On the contrary, the line containing the eye position and a point in the image plane above the vanishing line does not intersect the object plane. Hence, a vanishing line can be used to distinguish the region in the image corresponding to the object plane from that corresponding to the space above it. For example, if a horizon appears in the landscape image, the region below the horizon corresponds to the object plane.

3.2. Model construction

Our modeling scheme is different from the spidery mesh in that it is based on the vanishing line rather than the vanishing point in the image. With the new scheme, a vanishing line in the input image is interactively specified by a user, and the image is then divided by the vanishing line into two disjoint regions. The region below the vanishing line in the image corresponds to the object plane in the 3D environment, and that above the vanishing line corresponds to the space above the object plane. This space contains no objects that occlude another ones after all the foreground objects are extracted from the input image. Hence this space can be thought of as a plane of infinite distance. Based on this observation, we construct a background model with two planes (called a 'ground plane' and a 'back plane') corresponding to the two regions in the image separated by the vanishing line (Fig. 4).

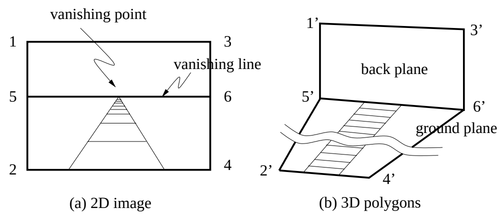

Figure 4: *new background model*

In Fig. 5, vertices $1-4$ are the four corners of the image and vertices 5 and 6 are the intersection points of the vanishing line with the image boundary. To render the background model, the 3D coordinates of the vertices of the model must be computed. For easy computation, we assume that the camera is positioned at the origin, the view direction is towards +z, the view-up vector is towards +y, and the focal length of the camera is d.

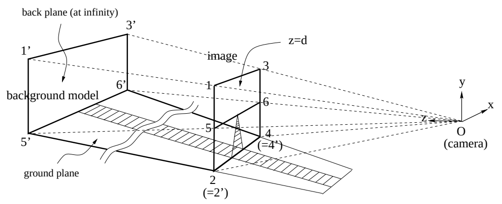

Figure 5: *Vertex coordinates of the new background model*

The vertices $1-6$ of the image have the following coordinates in the image plane:

$$\begin{array}{lll} 1:(x_L, y_1, d), & 2:(x_L, y_2, d), & 3:(x_R, y_3, d) \\ 4:(x_R, y_4, d), & 5:(x_L, y_5, d), & 6:(x_R, y_6, d) \end{array}$$

where x_L and x_R are the x-coordinates of the left and the right border of the image, respectively. Vertices $2'$ and $4'$ in the background model coincide with corner vertices 2 and 4, at the bottom of the image respectively. The rest of vertices, $1'$, $3'$, $5'$, and $6'$ are respectively the ideal points in the directions from the viewpoint to the corresponding vertices, 1, 3, 5, and 6 in the image. For example, vertex $3'$ is assigned a homogeneous coordinate $(x_R, y_3, d, 0)$ since this vertex is an ideal point in the direction from the origin to the vertex 3 in the image. Similarly, the homogeneous coordinates of each vertex in the background model are given. Thus, the coordinates of each vertex in the background model are:

$$\begin{array}{lll} 1':(x_L, y_1, d, 0), & 2':(x_L, y_2, d, 1), & 3':(x_R, y_3, d, 0) \\ 4':(x_R, y_4, d, 1), & 5':(x_L, y_5, d, 0), & 6':(x_R, y_6, d, 0) \end{array}$$

With these vertex coordinates, the equation for each plane of the background model can be obtained. The plane that contains a point $p = (x_1, x_2, x_3, x_4)$ satisfies

$$\pi_1 x_1 + \pi_2 x_2 + \pi_3 x_3 + \pi_4 x_4 = 0.$$

Let the coordinates of the three distinct points q, r, s be $(q_1, q_2, q_3, q_4), (r_1, r_2, r_3, r_4), (s_1, s_2, s_3, s_4)$, respectively. Then, the coefficients of the plane containing these points are computed as follows:

$$\pi_1 = \begin{vmatrix} q_2 & q_3 & q_4 \\ r_2 & r_3 & r_4 \\ s_2 & s_3 & s_4 \end{vmatrix}, \pi_2 = -\begin{vmatrix} q_1 & q_3 & q_4 \\ r_1 & r_3 & r_4 \\ s_1 & s_3 & s_4 \end{vmatrix},$$

$$\pi_3 = \begin{vmatrix} q_1 & q_2 & q_4 \\ r_1 & r_2 & r_4 \\ s_1 & s_2 & s_4 \end{vmatrix}, \pi_4 = -\begin{vmatrix} q_1 & q_2 & q_3 \\ r_1 & r_2 & r_3 \\ s_1 & s_2 & s_3 \end{vmatrix}.$$

Using the above equations, the 'ground plane' is computed from vertices $2'$, $4'$, $5'$ (or $6'$), and the 'back plane' is computed from vertices $1'$, $3'$, $5'$ (or $6'$). The resulting plane equations are used for constructing the scene model and for rendering, which will be discussed in Section 4, and 5, respectively.

3.3. Discussion

Based on projective geometry, our scheme is simple and yet general enough to cover a broader class of images than that of Horry et al. For a one-point perspective image, the left wall, the right wall, and the ceiling, constructed by the standard spidery mesh, can be modeled as foreground objects with our scheme. For a perspective image with two vanishing points, our scheme is still applicable since those points lie on a vanishing line. As mentioned in Section 2, the simplified spidery mesh of Horry et al. is used for an image that has no clearly identified vanishing point. This simplified spidery mesh looks similar to our modeling scheme since both of the resulting background models consist of only two quadrangles. That is, the rear wall and the floor of their model look similar to back and ground planes of our model, respectively.

However, the back plane is located at infinity together with the vanishing line, while the rear wall is placed at some place of finite distance from the original viewpoint to prevent the camera from going beyond it.

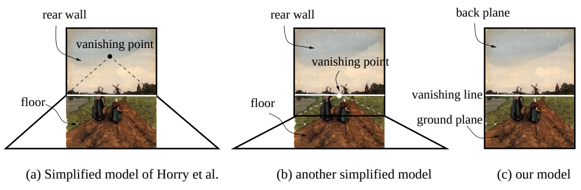

Figure 6: *Comparison between the simplified background model of Horry et al. and our model*

Fig. 6 shows an image which needs the simplified spidery mesh when the method of Horry et al is adopted. The horizon (i.e., the vanishing line) divides the image into two regions for both schemes. According to projective geometry, a vanishing point should lie somewhere on this vanishing line. However, as shown in the Fig. 6a, the vanishing point is not placed on the vanishing line with the simplified spidery mesh, which is a contradiction. Suppose that we place the vanishing point on the bottom edge of the rear wall (where the vanishing line is located in this image). Then, the floor would disappear in their background model. To avoid this, we have to place the vanishing line inside the rear wall as shown in Fig. 6b. Unfortunately, it causes the ground in the image to be divided into two perpendicular planes in 3D space. On the contrary, our background model is well suited for this image as shown in Fig. 6c.

4. Foreground model

A foreground object specified in the image is modeled as a 3D polygon, and the coordinates of its vertices are computed by finding the intersection points between the ground plane and the ray starting from the viewpoint and passing through the corresponding vertices in the image plane. For example, the foreground object in Fig. 7 is modeled as a quadrangle with four vertices P_0', P_1', P_2', and P_3'. These vertices have the corresponding vertices P_0, P_1, P_2, and P_3 in the image. Given a plane Π whose coefficients are $\pi_1, \pi_2, \pi_3, \pi_4$, and two arbitrary points $\vec{q}$ and $\vec{r}$, the intersection point $\vec{p}$ between Π and the line containing $\vec{q}$ and $\vec{r}$ is given as follows:

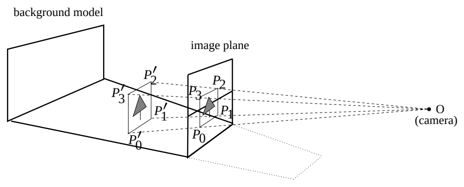

Figure 7: *Vertex coordinates of the foreground objects*

$$\vec{p} = (\vec{q} \cdot \vec{\pi}^T)\vec{r} - (\vec{r} \cdot \vec{\pi}^T)\vec{q} \tag{1}$$

where $\vec{\pi} = (\pi_1, \pi_2, \pi_3, \pi_4)$ [16]. This equation gives the coordinates of the vertices P_0' and P_1' (on the ground plane). That is, P_0' is the intersection of the ground plane with a line containing the viewpoint and P_0. P_1' is found similarly. The plane equation for the foreground polygon is then computed with an assumption that the polygon stands perpendicular to the ground plane. With the plane equation thus obtained, the coordinates for the remaining vertices P_2', P_3' are computed. By attaching all such polygons to the ground plane, the scene model is completed.

As proposed in [13], a foreground object can have a hierarchical structure in a more complex environment. That is, another foreground object can be attached on a foreground object standing on the ground plane to form a hierarchical structure. The wall and the ceiling generated by the standard spidery mesh can be modeled as a hierarchical foreground object with our scheme.

As mentioned in Section 2, the method of Horry et al. requires an input image and a foreground mask to render the foreground objects. Assuming that there is no foreground object occluding others, the input image gives the *RGB* color information of the foreground objects, and the foreground mask gives the alpha values specifying the exact portion of those objects. Instead of splitting the information in separate images, we use a single *RGBA* image called foreground image, which is generated by appending the alpha values of the foreground mask to the input image. That is, the foreground image consists of both the *RGB* components and the alpha values for the foreground objects. The alpha value of 1 is assigned inside the foreground object, and the value of 0 is assigned elsewhere to show the background model. To enable the smooth transition from the foreground to the background, a fractional value between 0 and 1 is assigned in the boundary of the foreground object. The foreground image thus obtained is texture-mapped onto the corresponding foreground polygon.

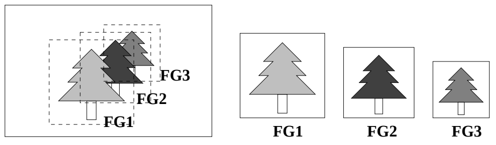

Figure 8: *Foreground objects and corresponding foreground images*

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. As suggested in [14], multiple foreground images are required in this case so that each occluded object can be associated with its corresponding foreground image. Each foreground image is obtained by recovering the occluded region of the corresponding foreground object using a conventional 2D painting tool. In addition, appropriate alpha values are assigned to specify the exact portion of the recovered foreground object. Each of these foreground images is texture-mapped on the corresponding foreground polygon. Fig. 8 shows an image where some foreground objects are occluded by others. Their corresponding foreground images are also shown on the right.

5. Rendering

An output image is generated by determining which part of the scene model is viewed by the camera through each pixel of the image. As shown in Fig. 9, we find the location of a point P on the scene model which is viewed through each pixel P_O of the output image from a given camera, by intersecting the viewing ray with the polygons in the scene model. Provided with the initial camera position and orientation, the point P_R in the reference image corresponding to the point P can be computed. The pixel P_O is then assigned the same color as that of the point P_R. To reduce the aliasing artifact, the color of P_R is decided by bilinearly interpolating those of the four nearest pixels. If P is on the background model, the background image is used as a reference image. Otherwise, P is on one of the foreground polygons. In this case, the alpha value of P_R in the corresponding foreground image is checked to see if P is inside an object in a foreground model. If the alpha value is 1, then P is inside the foreground object and the color of P_R is assigned to P_O. However, if the alpha value is less than 1, P is outside or on the boundary of the foreground object and additional checks are done to find the next intersection point P' in the scene model. Thus, the color of P_O is decided by blending the color of P and that of P' with the specified alpha value. This process is repeated until the next intersection point is either on a background model or on a foreground object whose alpha value is 1 at that point.

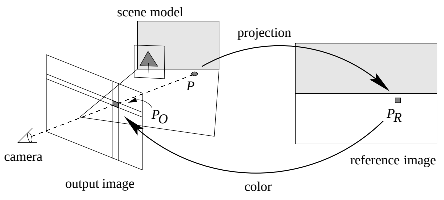

Figure 9: *Rendering mechanism*

However, it is time-consuming to find P and P_R for every pixel P_O in the output image. Instead, we use hardware texture mapping with OpenGL. Since an infinite value is not allowed in OpenGL, we should set the last component of the homogeneous coordinates for the back plane to have an infinitesimally small value. The background image is divided by the vanishing line into two subimages, which are texture-mapped onto the two polygons of the background model, respectively. For each of the foreground polygons, the corresponding foreground image is used as a texture map. As mentioned in the previous section, the appropriate alpha values in the foreground images are referenced to render the foreground objects correctly.

In hardware texture mapping, the graphics engine performs automatic perspective correction on the texture coordinates. In our case, however, the texture maps have already gone through the necessary perspective distortion. This means that simply assigning the texture coordinates to the corresponding vertices produces incorrect results caused by double perspective corrections. To avoid this, we transform the texture coordinates according to the depths of the corresponding vertices so that the automatic perspective correction is nullified. Details of this transformation is described in [6].

6. Extension to panoramic images

In this section, we show how to extend our modeling scheme to a panoramic image such as cylindrical or spherical image. Since the spherical image is regarded as the most general form of panoramic image, we focus on demonstrating our modeling scheme for a spherical image. For other types of panorama images, this scheme can easily be adapted.

6.1. Background model

In a (spherical) panorama image, the environment viewed from a camera is mapped onto the base sphere centered at the camera position. As shown in Fig. 10, parallel lines A and B on the ground plane are projected onto this sphere as arcs A' and B', respectively. A' and B' intersect at a point on the base sphere referred to as a vanishing point of the spherical image. Since A and B on the ground plane take on any inclinations, the set of all vanishing points on the sphere form a circle, that is the intersection of the base sphere with the horizon plane. This circle is said to be a vanishing circle that is analogous to the vanishing line for the planar image.

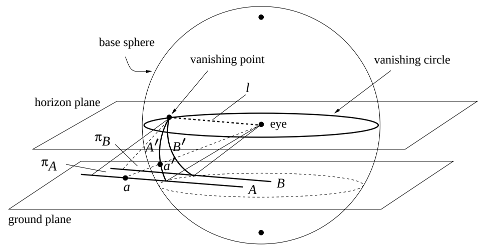

Figure 10: *Vanishing point and vanishing circle*

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 on a line connecting the viewpoint and a point on the vanishing circle.

Instead of the back plane in the background model for a planar image, we use a hemisphere of an arbitrarily large radius centered at the camera position. In analogy to the back plane, we call it the back hemisphere. We first specify the vanishing circle interactively and then project the upper hemisphere of the base sphere on the back hemisphere. The lower hemisphere is projected onto the ground plane. In practice, the back hemisphere is set to have some finite radius to avoid computational dificulty. This is equivalent to slightly moving down the vanishing circle from the horizon plane.

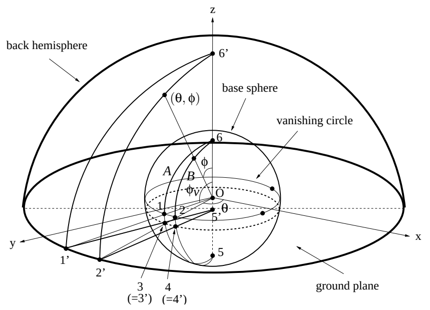

Figure 11: *Point correspondences between the base sphere and the background model*

The correspondence between the points on the base sphere and those on the background model is illustrated in Fig. 11. The base sphere is centered at the origin O and has two poles, that is, points 5 and 6. Consider great arcs A and B on the base sphere. Let points 1 and 2 be the intersections of A and B with the vanishing circle, respectively. Their intersections with the ground plane are points 3 and 4, respectively. Arc A contains points 6, 1 and 5, and arc B does points 6, 2, and 5. Points 1 and 2 are mapped to points $1'$ and $2'$ on the bottom circle of the back hemisphere. The points 3 and 4 are coincident with their projections, that is, points $3'$ and $4'$ on the ground plane. Points 5 and 6 are mapped to points $5'$ and $6'$. Point $5'$ is on the ground plane, and point $6'$ is on the back hemisphere. Thus, the region 6, 1, 2 on the base sphere are projected to the region $6', 1', 2'$ on the back hemisphere, the region 1,2,3,4 to the region $1', 2', 3', 4'$ on the ground plane, and finally the region 3,4,5 to region $3', 4', 5'$ on the ground plane.

Now, we show how to compute the position of the point on the background model, corresponding to a point on the base sphere. Without loss of generality, we assume that the camera is positioned at the origin, the initial view-up vector is towards +z, the horizon plane is coincident with the x-y plane, and the height of the camera from the ground plane is h. We use a spherical coordinate (θ, ϕ) for each point on the base sphere. Let ϕ_v denote the angle between the positive z-axis and the vector from the origin to a point on the vanishing circle. If $\phi < \phi_v$, then the projected point is on the back hemisphere; otherwise, it is mapped on the ground plane. The radius R of the back hemisphere is $-h/\cos\phi_v$. Therefore, a point (θ, ϕ) 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.

The background image for the background model is obtained in a similar way to that for the planar image. The foreground objects are interactively segmented out from the panoramic image and filled by inpainting techniques [17, 18, 19]. Again, 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 a hardware texture mapping, we have to make sure all the texture maps are for linear (line-to-line) perspective mapping. Since the spherical image contains a line-to-arc map, we cannot directly use this image as a texture map for the ground plane. Thus, aside from the transformation avoiding incorrect results caused by double perspective corrections (see Section 5), the lower subimage of a spherical image needs an additional transformation to be used as a texture map for the ground plane. That is, we convert the non-linear map into a linear map using a projective mapping from the base sphere onto the ground plane.

6.2. Foreground model

A foreground object for a panoramic image is interactively specified as a polygon standing on the ground plane. We compute the coordinates for the two vertices of the polygon on the ground plane. Those vertices form the bottom edge of the polygon on which it stands. Assuming that the foreground polygon stands vertically to the ground plane, the coordinates of the remaining vertices are computed using Equation 1 in Section 4. In Fig. 12, foreground objects A and B on the base sphere are respectively modeled as a polygonal object A' and B' standing on a ground plane. Note that A is modeled inside the base sphere since the bottom of the object appears under the ground plane. On the other hand, the object B, whose bottom falls between the horizon plane and the ground plane, goes outside the base sphere. The scene model is completed when all the foreground polygons are attached on the ground plane.

The foreground image for each foreground object is obtained from the base object. Similar to the background image, we cannot directly use the panoramic image since it is a non-linear texture map. By converting it into a linear map, we obtain the foreground image. To do this, we first set a bounding rectangle of the foreground polygon on a plane

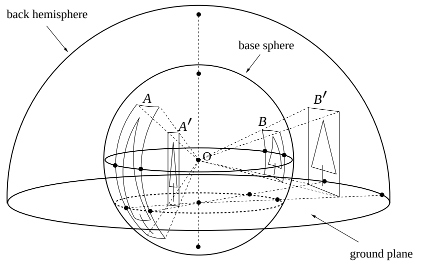

Figure 12: *Foreground model for a spherical image*

containing it and then project onto this rectangle the portion of the panoramic image which 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 image of the object. The exact portion of the foreground object is determined by assigning appropriate alpha values after interactive segmentation of the foreground image. With all the texture images thus obtained, we generate a novel view from a desired camera position by rendering the constructed scene model.

7. Experimental Results

Fig. 13 illustrates how to generate an image viewed from a new viewpoint. Given an input image (Fig. 13a), the vanishing line and the foreground objects are interactively specified (Fig. 13b). Based on the specification, the 3D scene model is constructed and the texture images such as those in Figs. 13c and 13d are prepared. Fig. 13e is an image generated after moving the camera forward from its initial position in the scene. Note that only the foreground objects are scaled, leaving the background fixed. This type of result cannot be obtained just by scaling the image.

Fig. 14b shows the individual foreground images whose occluded regions are recovered. Each of these images is to be texture-mapped onto the corresponding foreground object. The pixels not covered by a foreground object are marked with an alpha value of 0 in the foreground image so that they do not appear in the output image. Even a new object that does not exist in the input image can be inserted into the scene model with its foreground image. Fig. 14c is the scene image obtained after moving the camera close to the foreground objects. As shown in this figure, the occluded portion of a house is recovered (the second foreground image), and a man who does not appear in the input image, is added as a foreground object beside the front house (the last foreground image).

Fig. 15 illustrates a case with an image containing two vanishing points. These vanishing points are formed by two distinct roads in the scene. Although Horry et al. proposed a modeling scheme for an image with two vanishing points, it is not suitable for this image since their scheme only deals with the case where the two vanishing points are formed by a box-shaped structure in the image. Their standard spidery mesh is not suitable either since only one of the two vanishing points should be selected to construct the background model, resulting that the other one appears on the left or the right wall.

In Fig. 16, our modeling scheme is compared with that of Horry et al. by applying them to the same picture. As shown in Fig. 16a, the vanishing point is clearly identified in this input image. When the scheme of Horry et al. is applied to this image, the vanishing point is placed inside the rear wall and thus a straight line (the railroad in the input image) bends as the camera moves leftward or rightward (Fig. 16b). However, the straight line is preserved with our scheme since the image is divided by the vanishing line so that the ground plane or the back plane is prevented from being separated into multiple planes. Fig. 16c shows our model viewed from the same camera position as that of the Fig. 16b.

Fig. 17 shows the result of our scheme applied to a picture which was used in [14]. The scene in the Fig. 17a is well suited for the scheme of Horry et al. since only one vanishing point is formed by the road in the image, and the buildings are placed on both sides of the road. Their scheme (i.e., the standard spidery mesh) constructs the background model by assigning the left buildings in the image to the left wall, the right buildings to the right wall, the building in the middle to the rear wall, the ground to the floor, and the sky to the ceiling. All the remaining objects are modeled as foreground objects. On the other hand, our scheme constructs the background model just with the ground and the sky in the image, and all the buildings and other objects are considered as foreground objects (Fig. 17b). Yet, a similar result to that of Horry et al. can be generated with our scheme as shown in Fig. 17c.

Fig. 18 demonstrates our extended modeling scheme applied to a spherical panoramic image. Given an input spherical image (Fig. 18a), the scene model is constructed based on the location of a vanishing circle (Fig. 18b). Fig. 18c $\sim$ e are some output images obtained from novel viewpoints other than the center of the base sphere. The 3D parallax effects shown in these images cannot be generated by typical panoramic image viewers. Similarly, results for a cylindrical input image are shown in Fig. 19. Unlike for the spherical image, there appears a hole at the center of ground plane since the cylindrical image has no information for this region. This hole, however, can be easily recovered by inpainting techniques since the corresponding spot in the scene tends to be flat and contains no important objects, considering that the photographer stood on this spot for taking pictures.

The rendering speed is dependent on the number of foreground objects in the image and the image size. Our system is implemented in C++ with OpenGL library on Intel Pentium^{R} PC (PIII 800 MHz processor and 512 MB memory) equipped with nVIDIA GeForce2GTS^R graphics processor. On average, the output sequence of images with 500×400 pixels is generated at an interactive rate (over 100 frames/sec).

8. Conclusions and Future work

In this paper, we proposed a new modeling scheme for TIP based on a vanishing line. We first divide the input image into two regions by the vanishing line, and then construct the scene model consisting of two planes corresponding to those regions in the image. According to projective geometry, these planes represent the background of infinite distance and the ground on which the viewer stands, respectively. Our scheme is simpler than the spidery mesh of Horry et al., and yet more general to cover a broader class of input images.

We also have shown our modeling scheme is naturally extended to navigation into a panoramic image. Based on a vanishing circle which is analogous to the vanishing line in a planar image, the scene model for a panoramic image is interactively constructed, which is then used for navigation. Compared to the conventional panoramic viewers (such as QuickTime VR^R) which allows camera pan, tilt, and zoom control from only a fixed viewpoint, we provide the real sense of walk-through or navigation into the panoramic scene by enabling continuous camera translation as well as rotation.

We are currently working on extending our scheme to a video stream, that is, a sequence of images. Exploiting time coherence, we believe that our scheme can be enhanced to handle an interesting class of scenes in videos such as sports games and cartoon animations that have a static background and some moving foreground objects.

9. Acknowledgements

This work has been supported by the NRL (National Research Laboratory) project of KISTEP (Korea Institute of Science & Technology Evaluation and Planning).

References

1. A. Lippman. “Movie Maps: An Application of the Optical Videodisc to Computer Graphics”, *ACM SIGGRAPH 80 Conference Proceedings*, pp. 32–43 (August 1980). 2
2. A. R. Mohl. “Cognitive Space in the Interactive Movie Map: an Investigation of Spatial Learning in the Virtual Environments”, PhD Thesis, MIT, 1981. 2
3. G. Miller, E. Hoffert, S. E. Chen, E. Patterson, D. Blackketter, S. Rubin, S. A. Applin, D. Yim, and J. Hanan. “The Virtual Musuem: Interactive 3D Navigation of a Multimedia Database”, *The Journal of Visualization and Computer Animation*, **3**, pp. 183–197 (August 1992). 2
4. S. E. Chen. “QuickTime VR - An Image-Based Approach to Virtual Environment Navigation”, *ACM SIGGRAPH 95 Conference Proceedings*, pp. 29–38 (August 1995). 2
5. S. E. Chen and L. Williams. “View Interpolation for Image Synthesis”, *ACM SIGGRAPH 93 Conference Proceedings*, pp. 279–288 (August 1993). 2
6. L. Darsa, B. C. Silva, and A. Varshney. “Navigating Static Environments Using Image-Space Simplification and Morphing”, *Proc. 1997 Symposium on Interactive 3D Graphics*, pp. 25–34 (April 1997). 2, 6
7. L. McMillan and G. Bishop. “Plenoptic Modeling: An Image-Based Rendering System”, *ACM SIGGRAPH 95 Conference Proceedings*, pp. 39–46 (August 1995). 2
8. W. R. Mark and L. McMillan and G. Bishop. “Post-Rendering 3D Warping”, *Proceedings 1997 Symposiums on Interactive 3D Graphics*, pp. 7–16 (1997). 2
9. M. Levoy and P. Hanrahan. “Light Field Rendering”, *ACM SIGGRAPH 96 Conference Proceedings*, pp. 31–42 (August 1996). 2
10. S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen. “The Lumigraph”, *ACM SIGGRAPH 96 Conference Proceedings*, pp. 43–54 (August 1996). 2
11. P. -P. Sloan, M. F. Cohen, and S. J. Gortler. “Time Critical Lumigraph Rendering”, *Proceedings 1997 Symposium on Interactive 3D Graphics*, pp. 17–23 (1997). 2
12. P. E. Debevec, C. J. Taylor, and J. Malik. “Modeling and rendering architecture from photographs: A hybrid geometry-and image- based approach”, *ACM SIGGRAPH 96 Conference Proceedings*, pp. 11–20 (August 1996). 2
13. Y. Horry, K. Anjyo, and K. Arai. “Tour Into the Picture: Using a Spidery Mesh Interface to Make Animation from a Single Image”, *ACM SIGGRAPH 97 Conference Proceedings*, pp. 225–232 (August 1997). 1, 5
14. K. Anjyo and Y. Horry. *Theory and Practice of “Tour Into the Picture”*. In *Course Notes #8 (SIGGRAPH ’98)*, 1998. 3, 5, 8
15. D. Liebowitz, A. Criminisi, and A. Zisserman. “Creating Architectural Models from Images”, *EuroGraphics 99 Conference Proceedings*, pp. 39–50 (September 1999). 2
16. M. A. Penna and R. R. Patterson. *Projective Geometry and Its Applications to Computer Graphics*. Prentice-Hall, 1986. 1, 5
17. A. A. Efros and T. K. Leung. “Texture Synthesis by Non-parametric Sampling”, *Proceedings of IEEE International Conference on Computer Vision*, (1999). 7
18. M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester. “Image Inpainting”, *ACM SIGGRAPH 2000 Conference Proceedings*, pp. 417–424 (August 2000). 7
19. L.-Y. Wei, M. Levoy. “Fast Texture Synthesis using Tree-structured Vector Quantization”, *ACM SIGGRAPH 2000 Conference Proceedings*, pp. 479–488 (August 2000). 7

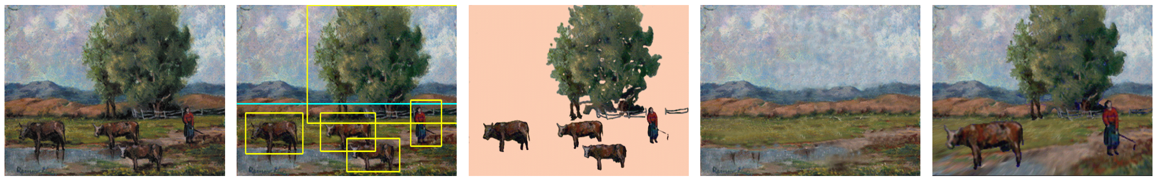
(a) Input image (b) User's specification (c) Foreground image (d) Background image (e) Output image

Figure 13: *Generating a scene from a new camera position*

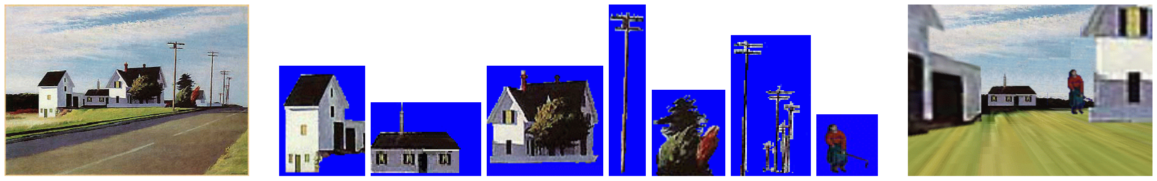
(a) Initial scene (b) Foreground images for the foreground objects (c) Moving forward

Figure 14: *Multi−layered foreground objects*

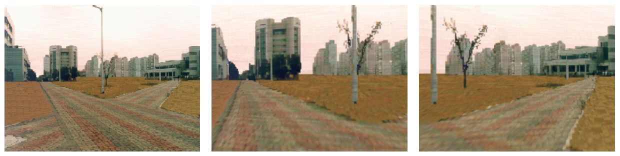
(a) Initial scene (b) Moving toward the left vanishing point (b) Moving toward the right vanishing point

Figure 15: *Image with multiple vanishing points*

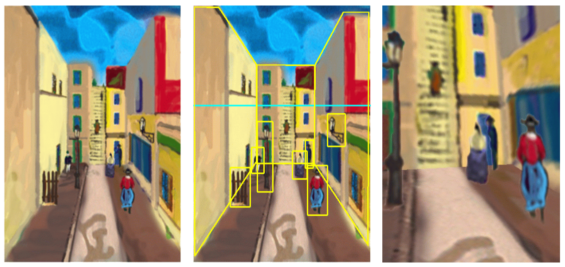
(a) Input image (b) Specification (c) Moving forward

Figure 17: *One−point perspective image*

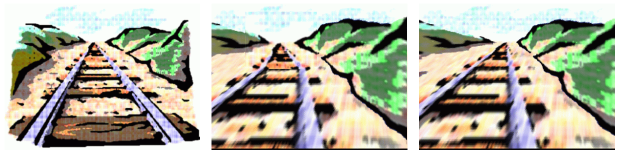
(a) Input image (b) Model of Horry et al. (c) Our model

Figure 16: *Image distortion*

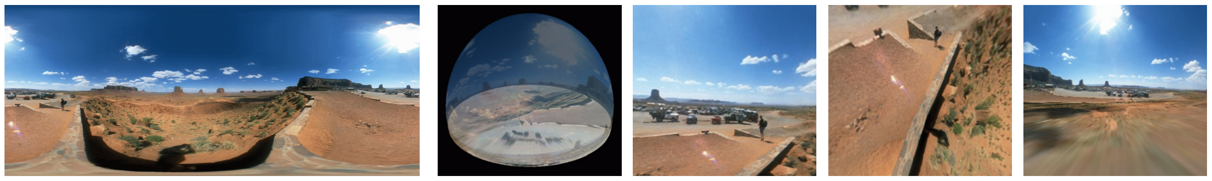
(a) Spherical image (b) Scene model (c) Output image 1 (d) Output image 2 (e) Output image 3

Figure 18: *Spherical panoramic image*

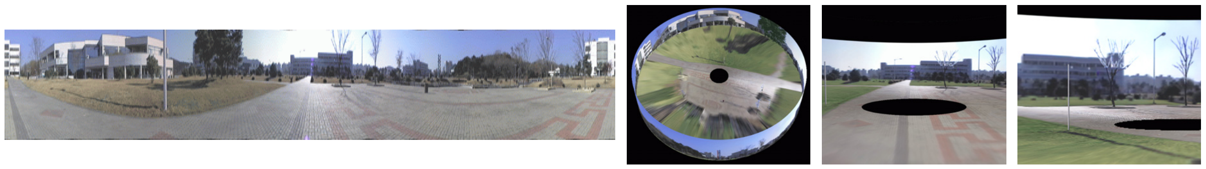
(a) Cylindrical image (b) Scene model (c) Output image 1 (d) Output image 2

Figure 19: *Cylindrical panoramic image*