are treated with different weightings in an attempt to make the optimization more stable.

Most IBMR techniques deal with photographs or synthesized images and require two or more images as input. Creating an animation from an existing painting is sometimes considered more difficult then from a photograph because a painting does not provide as precise information for creating a 3D scene model as a photograph does. The TIP technique [HAA97] is one of the few that can generate novel views from a single input image of painting or drawing, as well as photograph. Fleishman *et al.* [FCK+99] attempted to use the TIP technique to link several sparse views that are taken along a specific camera path. This approach removes the problem of limited touring space of the TIP technique, but the image correspondence problem was not solved satisfactorily to produce good walk-through animations. Kang *et al.* [KAS01] extended the TIP technique to utilize vanishing lines and applied it onto panoramic images; however, only the abstract is available at present. At the time of writing this thesis, we obtained a pre-print version of a conference paper by Oh *et al.* [OCD+01], which presents an image-based modeling and photo editing system, in which the use of *ground plane* and *vertical tool* bears certain resemblance with our way of modeling vertical shapes, which is described in Chapter 4.

2.2 Review of the TIP technique

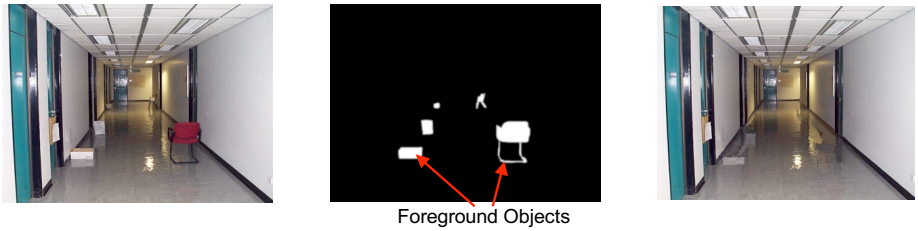
In this section, we give a review of the *Tour Into the Picture (TIP)* technique [HAA97] on which our modeling approaches are based. Here, we only describe the case of one-point perspective. The main difference between one-point and two-point perspectives in TIP is in the way we derive the background model and the latter case is described in Appendix A. Pictures with three-point perspective are rare, so in our discussion we restrict ourselves to the former two cases.

The TIP technique allows users to make animations from a single picture or photograph of a scene by building a simple model for the scene. The scene model constructed is not exact; it comprises only a small number of polygons in 3D space. Before modeling, the user is required to prepare two supporting images from the original image. After these two images are imported into the TIP system, a simple

scene model is constructed by the user using a special graphical user interface. Then, novel views of the 3D model are rendered via 2D to 3D transformations. These procedures are discussed in more details below.

2.2.1 Pre-processing

In the pre-processing phrase, two new images, the *foreground mask* and the *background image* (see Figure 2-1), are prepared from the original image using commercial software such as *Photoshop* [Pho2]. In the foreground mask, objects identified by user as foreground objects are colored white while the background is filled with black. The foreground mask also serves as the alpha channel for the texture of the foreground objects and gradual shades at object boundaries are possible. With interactive segmentation techniques such as intelligent scissors [MB95b] implemented as lasso tools in commercial software, it is not difficult to segment the foreground objects. The background image is obtained by erasing the foreground objects and patching the missing background details using tools like 'clone brush' in commercial software.



(a) Original picture (b) Foreground mask (c) Background image

Figure 2-1. Foreground mask and background image obtained from the original picture.

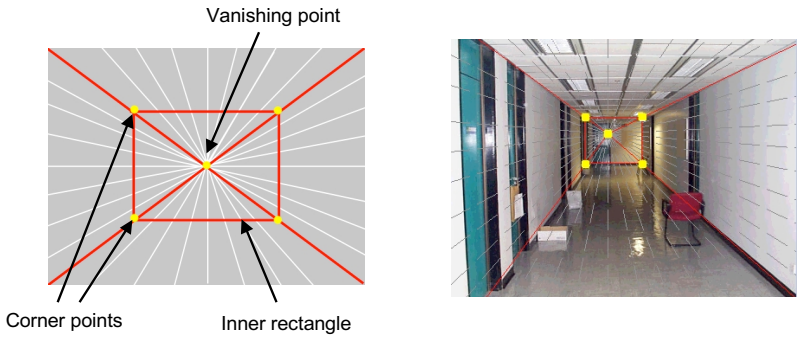
2.2.2 Modeling

The modeling process can be divided into two parts: background modeling and foreground object modeling.

Background modeling

The user is presented with a special graphical user interface (GUI) called the *spidery mesh* as shown in Figure 2-2(a). This spidery mesh consists of a vanishing point, an inner rectangle (which is intuitively the window out of which we look to the

infinity), and radial lines that radiate from the vanishing point. Each side of the inner rectangle is parallel to a side of the input image frame. The spidery mesh is used in the following way. There are essentially five control points which the user can manipulate to fit the mesh to the perspective of scene: the four corner points of the inner rectangle, and the vanishing point. The default configuration of the spidery mesh is shown in shown in Figure 2-2(a). As the user drags a control point, radial lines emitting from the vanishing point will change accordingly. Figure 2-2(b) shows a spidery mesh fitted to an input image.



(a) Spidery mesh (b) Mesh fitted to an image

Figure 2-2. Using the spidery mesh GUI.

Five regions can be deduced from the specification of a spidery mesh, each of which is a 2D polygon as shown in Figure 2-3. The five regions are called *floor, ceiling, right wall, left wall,* and *rear wall*. A simple 3D box model consisting of five rectangles is then derived according to the pixel coordinates of the vertices of these polygons. We name each rectangle of the box model the same as its corresponding polygon on the mesh. The computation of the 3D coordinates of the 3D model is discussed below.

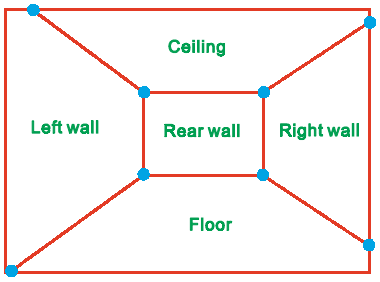
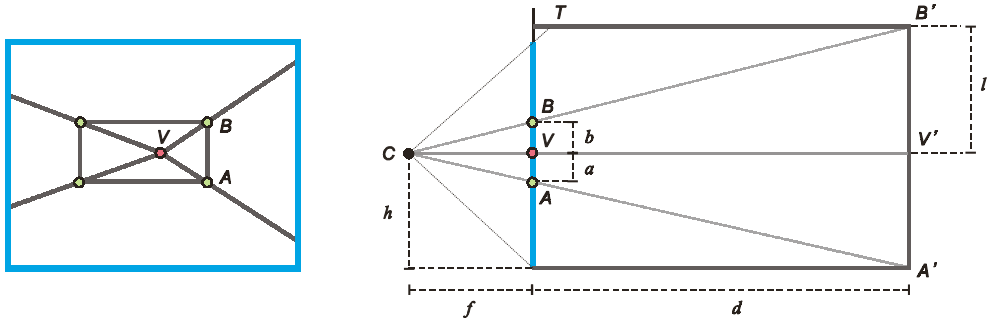


Figure 2-3. Modeling the background as five polygons.

Calculation 3D positions from a spidery mesh

Here, the model derived from the mesh is only up to a scale because scale is not important in this application. In fact, it is actually not possible to obtain the scale without a given measuring reference from the input image.



(a) Spidery mesh configuration (b) Side-view of the model to be derived

Figure 2-4. Deriving the box model from spidery mesh.

In TIP, the rear wall of the box model is assumed to be parallel to the image plane and the up-vector of the camera is parallel to the left and right walls of the box model. Since scale is irrelevant, the bottom of the image frame can be set to meet the floor rectangle of the background model. Figure 2-4(a) shows the configuration of a spidery mesh, and Figure 2-4(b) shows the side view of the image plane and the model to be derived from it. We use the following equivalent computation in our implementation. We denote the camera position as C, the distance of the camera from the projection plane as f, the vanishing point on the image plane as V, and the height of the camera above the floor rectangle as h. By perspective geometry, C must lie on the line that passes through V and perpendicular to the image plane. If we know f, we can then fix C. However, in one-point perspective view, the information given by the spidery mesh is insufficient for determining f (a minor GUI improvement for better visual estimation of f is presented in Chapter 5). In our derivation, we assume f to be equal to the maximum perpendicular distances of V to the four edges of the image frame, so that both the vertical and horizontal field-of-view's will not be beyond 90 degrees for all cases of mesh configuration.

With the above assumptions, the vertices of the background rectangles in 3D space can be derived by principles of similar triangles. Since we have already fixed

the floor of the box model to be at the plane $y = -h$ with respect to the camera coordinate system, the configuration of the box model can be completely determined if we can find its extents from *C* along the three axis of a coordinate frame that is aligned with the edges of the box model. To find *d* the depth of the box model, we can project *A* on to the ground plane as shown in Figure 2-4(b). Considering similar triangles ΔCV'A' and ΔCVA, we have

$$\frac{h}{f+d} = \frac{a}{f}$$

which gives

$$d = \frac{h \cdot f}{a} - f$$

Having found *d*, we can project *B* on to the plane of the rear wall to get the height of the box model. Considering similar triangles ΔCV'B' and ΔCVB, we have

$$\frac{l}{f+d} = \frac{b}{f}$$

which yields

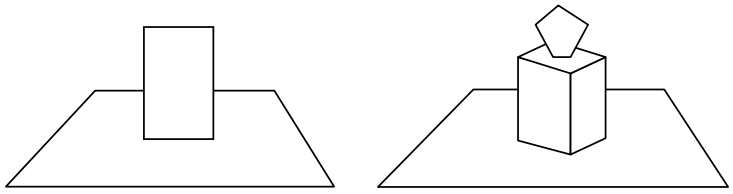
$$l = \frac{b(f+d)}{f}$$

The height of the box can then be taken as $(h+l)$. The extents of the box along its width can be calculated similarly and the box model will be completely specified when we have all its extents along its three dimensions.

If the vanishing point *V* is not at the center of the image, some of the walls of the box model will not have complete texture coverage. An example is shown in Figure 2-4(b), where the top edge of the image is projected to the point *T* on the ceiling of the box model, and there is no texture information for the ceiling near its front edge. This actually will not be an issue as we could just leave the missing part of the texture blank. But to save texture memory, we could also find the point *T* by projecting the top edge of the image frame onto the ceiling and crop away the blank part of the ceiling rectangle. The same could be applied to the other walls of the box model.

Foreground modeling

Foreground objects are modeled as 'billboards', which are just simple polygons attached to either the floor, ceiling, or one of the walls, as shown in Figure 2-5(a). To specify a foreground object, the user is required to draw a bounding polygon to enclose it. Horry *et al.* also proposed the use of hierarchical model for more complex foreground objects (see Figure 2-5(b)). For both simple and hierarchical models, each polygon is specified with the assumption that it is perpendicular to its parent polygon, so that its vertices in 3D space can be found by projecting the vertices of the user-specified bounding polygon on to some known planes. The derivation of the 3D positions of the billboards is similar to that of the background model.



(a) A single billboard (b) A hierarchical object

Figure 2-5. Foreground objects modeled as billboards.

2.2.3 Rendering

The original rendering method is basically to project the pixels to be rendered in novel views onto the surfaces of the 3D model, and then obtain the color and alpha values by re-projecting the points on the 3D model back to its initial 2D projection on the input images. Horry *et al.* do not first extract the texture for each background wall and foreground object from the input images and then utilize hardware texture mapping function during rendering, which is what we implemented in our animation system described in Chapter 3.