26

Application: Image-Based Rendering

The entertainment industry touches hundreds of millions of people every day, and synthetic pictures of real scenes, often mixed with actual film footage, are now common place in computer games, sports broadcasting, TV advertising, and feature films. Creating these images is what *image-based rendering*—defined here as the synthesis of new views of a scene from prerecorded pictures—is all about, and it does require the recovery of quantitative (although not necessarily three-dimensional) shape information from images. This chapter presents a number of representative approaches to image-based rendering, dividing them, rather arbitrarily, into (a) techniques that first recover a three-dimensional scene model from a sequence of pictures, then render it with classical computer graphics tools (naturally, these approaches are often related to stereo and motion analysis); (b) methods that do not attempt to recover the camera or scene parameters, but construct instead an explicit representation of the set of all possible pictures of the observed scene, then use the image position of a small number of tie points to specify a new view of the scene and *transfer* all the other points into the new image, in the photogrammetric sense already mentioned in chapter 10; and (c) approaches that model images by a two-dimensional set of light rays (or more precisely by the value of the radiance along these rays) and the set of all pictures of a scene by a four-dimensional set of rays, the *light field* (Figure 26.1).

26.1 CONSTRUCTING 3D MODELS FROM IMAGE SEQUENCES

This section addresses the problem of building and rendering a three-dimensional object model from a sequence of pictures. It is, of course, possible to construct such a model by fusing registered depth maps acquired by range scanners as described in chapter 21, but we focus here on the case where the input images are digitized photographs or film clips of a rigid or dynamic scene.

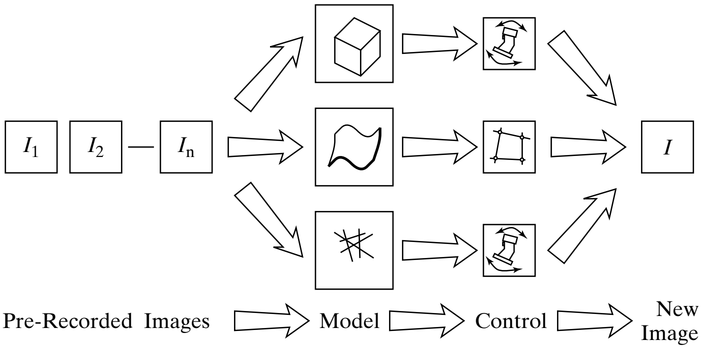

Figure 26.1 Approaches to image-based rendering. From top to bottom: three-dimensional model construction from image sequences, transfer-based image synthesis, the light field. From left to right, the image-based rendering pipeline: A scene model (that may not be three-dimensional) is constructed from sample images, and used to render new images of the scene. The rendering engine may be controlled by a joystick (or equivalently by the specification of camera parameters) or, in the case of transfer-based techniques, by setting the image position of a small number of tie points.

26.1.1 Scene Modeling from Registered Images

Volumetric Reconstruction Let us assume that an object has been delineated (perhaps interactively) in a collection of photographs registered in the same global coordinate system. It is impossible to uniquely recover the object shape from the image contours since, as observed in chapter 19, the concave portions of its surface never show up on the image contour. Still, we should be able to construct a reasonable approximation of the surface from a large enough set of pictures. There are two main global constraints imposed on a solid shape by its image contours: (a) it lies in the volume defined by the intersection of the viewing cones attached to each image, and (b) the cones are tangent to its surface (there are other local constraints; e.g., as shown in chapter 19, convex [resp. concave] parts of the contour are the projections of convex [resp. saddle-shaped] parts of the surface). Baumgart exploited the first of these constraints in his 1974 PhD thesis to construct polyhedral models of various objects by intersecting the polyhedral cones associated with polygonal approximations of their silhouettes. His ideas have inspired a number of approaches to object modeling from silhouettes, including the technique presented in the rest of this section (Sullivan and Ponce, 1998) that also incorporates the tangency constraint associated with the viewing cones. As in Baumgart's system, a polyhedral approximation of the observed object is first constructed by intersecting the visual cones associated with a few photographs (Figure 26.2). The vertices of this polyhedron are then used as the control points of a smooth *spline surface*, which is deformed until it is tangent to the visual rays. We focus here on the construction and deformation of this surface.

Spline Construction. A *spline curve* is a piecewise-polynomial parametric curve that satisfies certain smoothness conditions. For example, it may be C^k (i.e., differentiable with continuous derivatives of order up to k), with k usually taken to be 1 or 2, or G^k (i.e., not necessarily differentiable everywhere, but with continuous tangents in the G^1 case and continuous curvatures in the G^2 case). Spline curves are usually constructed by stitching together *Bézier arcs*. A Bézier curve of degree n is a polynomial parametric curve $P : [0, 1] \rightarrow \mathbb{E}^3$ defined as the barycentric

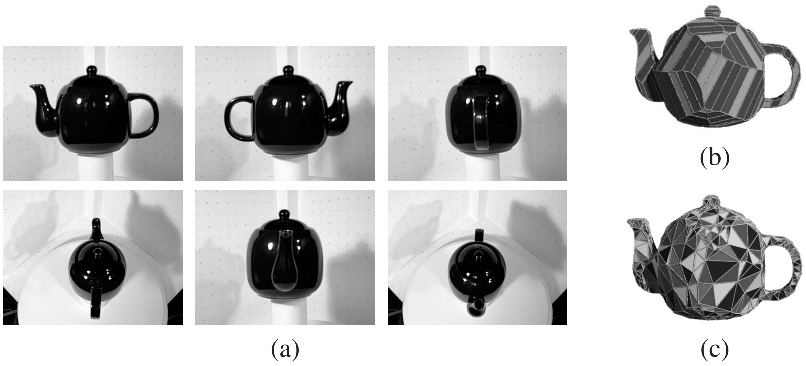

Figure 26.2 Constructing object models by intersecting (polyhedral) viewing cones: (a) six photographs of a teapot, (b) the raw intersection of the corresponding viewing cones, (c) the triangulation obtained by splitting each face into triangles and simplifying the resulting mesh. *Reprinted from "Automatic Model Construction, Pose Estimation, and Object Recognition from Photographs Using Triangular Splines," by S. Sullivan and J. Ponce, IEEE Transactions on Pattern Analysis and Machine Intelligence, 20(10):1091–1096, (1998). © 1998 IEEE.*

combination

$$P(t) = \sum_{i=0}^{n} b_i^{(n)}(t) P_i$$

of $n+1$ *control points* $P_0, \dots, P_n$, where the weights $b_i^{(n)}(t) \overset{\text{def}}{=} \binom{n}{i} t^i (1-t)^{n-i}$ are called the *Bernstein polynomials* of degree n.[1] A Bézier curve interpolates its first and last control points, but not the other ones (Figure 26.3a). As shown in the exercises, the tangents at its endpoints are along the first and last line segments of the *control polygon* formed by the control points.

The definition of Bézier arcs and spline curves naturally extends to surfaces: A triangular Bézier patch of degree n is a parametric surface $P : [0,1] \times [0,1] \to \mathbb{E}^3$ defined as the barycentric combination

$$P(u,v) = \sum_{i+j+k=n} b_{ijk}^{(n)}(u, v, 1-u-v) P_{ijk}$$

of a triangular array of control points P_{ijk}, where the homogeneous polynomials $b_{ijk}^{(n)}(u,v,w) \overset{\text{def}}{=} \frac{n!}{i!j!k!} u^i v^j w^k$ are the *trivariate Bernstein polynomials* of degree n. In the rest of this section, we use *quartic* Bézier patches ($n = 4$), each defined by 15 control points (Figure 26.3b). Their boundaries are the quartic Bézier curves $P(u,0)$, $P(0,v)$, and $P(u, 1-u)$. By definition, a G^1 *triangular spline* is a network of triangular Bézier patches that share the same tangent plane along their common boundaries. A necessary (but not sufficient) condition for G^1 continuity is that all control points surrounding a common vertex be coplanar. We first construct these points, then place the remaining control points to ensure that the resulting spline is indeed G^1 continuous. As discussed in Loop (1994), a set of coplanar points $Q_1, \dots, Q_p$ can be created as a barycentric

[1] This is indeed a barycentric combination (as defined in chapter 12) since the Bernstein polynomials are easily shown to always add to 1. In particular, Bézier curves are affine constructs—a desirable property since it allows the definition of these curves purely in terms of their control points and independently of the choice of any external coordinate system.

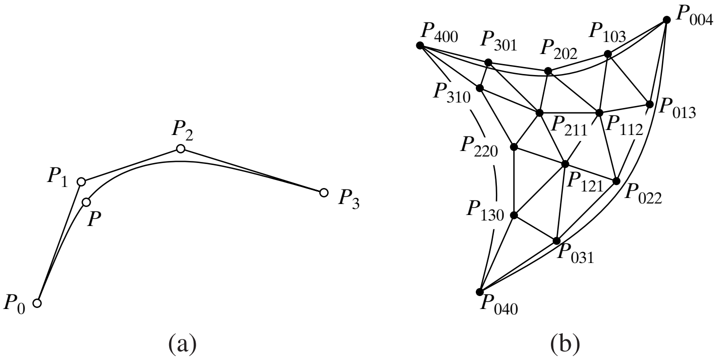

Figure 26.3 Bézier curves and patches: (a) a cubic Bézier curve and its control polygon; (b) a quartic triangular Bézier patch and its control mesh. *Tensor-product* Bézier patches can also be defined using a rectangular array of control points (Farin 1993). Triangular patches are, however, more appropriate for modeling free-form *closed* surfaces.

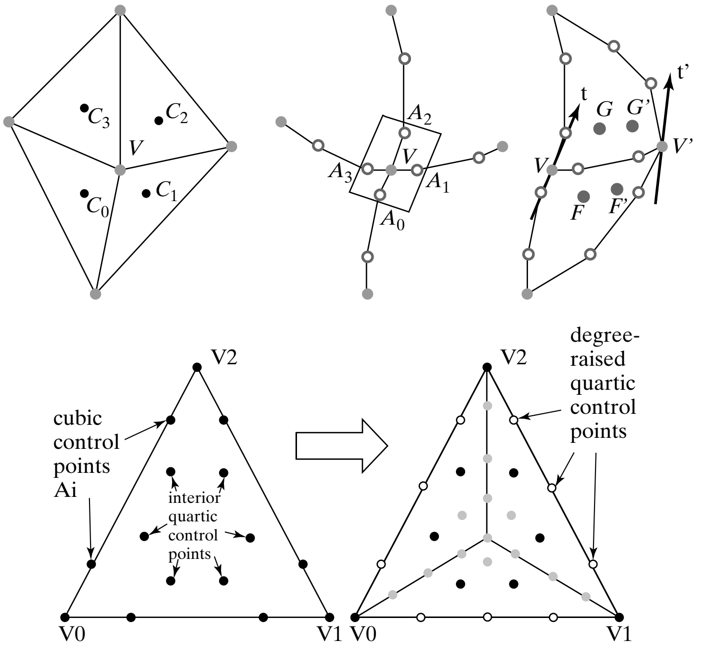

Figure 26.4 Construction of a triangular spline over a triangular polyhedral mesh. Top, from left to right: The cubic boundary control points, the boundary curves surrounding a mesh vertex, and the construction of internal control points from tangent specification. Bottom: Splitting a patch three ways to enforce G^1 continuity: The white points are the control points obtained by raising the degree of the control curves, and the gray points are the remaining control points, computed to ensure G^1 continuity. After Sullivan and Ponce (1998).

combination of p other points $C_1, \dots, C_p$ in general position (in our case, the centroids of the p triangles T_j adjacent to a vertex V of the input triangulation, Figure 26.4, top left) as

$$Q_i = \sum_{j=1}^{p} \frac{1}{p} \left\{ 1 + \cos\frac{\pi}{p} \cos\left([2(j-i)-1]\frac{\pi}{p} \right) \right\} C_j.$$

This construction places the points Q_i in a plane passing through the centroid O of the points C_i. Translating this plane so that O coincides with V yields a new set of points A_i lying in a plane passing through V (Figure 26.4, top center).

Since cubic Bézier curves are defined by four points, we can interpret two adjacent vertices V and V' and the points A_i and A'_i associated with the corresponding edge as the control points of a cubic curve. This yields a set of cubic arcs that interpolate the vertices of the control mesh and form the boundaries of triangular patches. Once these curves have been constructed, the control points on both sides of a boundary can be chosen to satisfy interpatch G^1 continuity. In this construction, the cross-boundary tangent field linearly interpolates the tangents at the two endpoints of the boundary curve. At the endpoint V, the tangent $\boldsymbol{t}$ across the curve that contains the point A_i is taken to be parallel to the line joining A_{i-1} to A_{i+1}. The tangent $\boldsymbol{t}'$ is obtained by a similar construction. The interior control points F, F', G, and G' (Figure 26.4, top right) are constructed by solving the set of linear equations associated with this geometric condition (Chiyokura, 1983). However, there are not enough degrees of freedom in a quartic patch to allow the simultaneous setting of the interior points for all three boundaries. Thus, each patch must be split three ways, using, for example, the method of Shirman and Sequin (1987) to ensure continuity among the new patches: Performing *degree elevation* on the boundary curves replaces them by quartic Bézier curves with the same shape (see Exercises). Three quartic triangular patches can then be constructed from the boundaries as shown in Figure 26.4, bottom. The result is a set of three quartic patches for each mesh face, which are G^1 continuous across all boundaries.

Spline Deformation. We have given a method for constructing a G^1-continuous triangular spline approximation of a surface from a triangulation such as the one shown in Figure 26.2(b). Let us now show how to deform this spline to ensure that it is tangent to the viewing cones associated with the input photographs. The shape of the spline surface S is determined by the position of its control vertices $V_1, \dots, V_p$. We denote by V_{jk} ($k = 1, 2, 3$) the coordinates of the point V_j ($j = 1, \dots, p$) in some reference Euclidean coordinate system, and use these $3p$ coefficients as shape parameters. Given a set of rays $R_1, \dots, R_q$, we minimize the energy function

$$\frac{1}{q}\sum_{i=1}^{q} d^2(R_i, S) + \lambda \sum_{i=1}^{r} \iint \left[|P_{uu}|^2 + 2|P_{uv}|^2 + |P_{vv}|^2 \right] du\, dv$$

with respect to the parameters V_{jk} of S. Here, $d(R, S)$ denotes the distance between the ray R and the surface S, the integral is a *thin-plate* spline energy term used to enforce smoothness in areas of sparse data, and λ is a constant weight introduced to balance the distance and smoothness terms. The variables u and v in this integral are the patch parameters, and the summation is done over the r patches that form the spline surface. The signed distance between a ray and a surface patch can be computed using Newton's method. For rays that do not intersect the surface, we define $d(R, S) = \min\{|\overrightarrow{QP}|, Q \in R, P \in S\}$, and compute the distance by minimizing $|\overrightarrow{QP}|^2$. For those rays that intersect the surface, we follow Brunie, Lavallée, and Szeliski (1992) and measure the distance to the *farthest* point from the ray that lies on the surface in the direction of the surface normal at the corresponding occluding contour point. In both cases, Newton iterations are initialized from a sampling of the surface S. During surface fitting, the spline is deformed

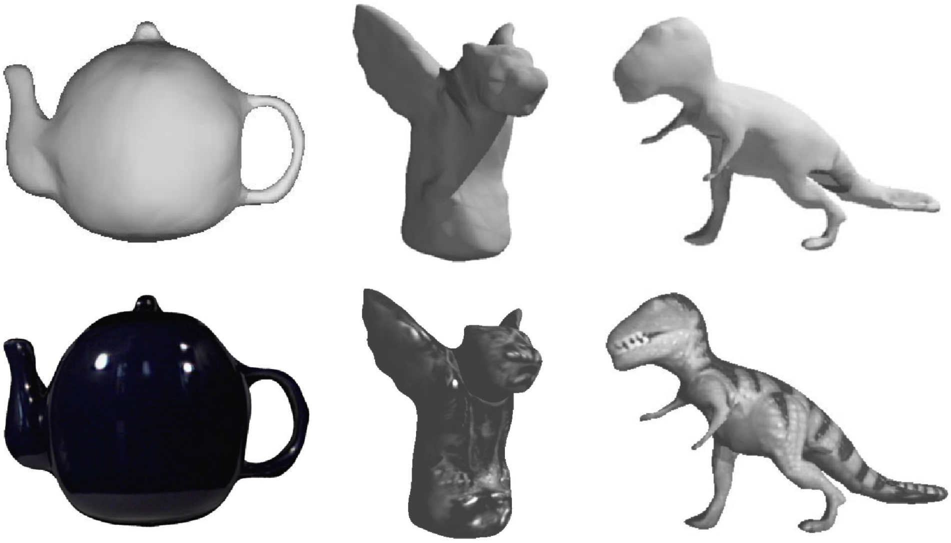

Figure 26.5 Shaded and texture-mapped models of a teapot, gargoyle and dinosaur. The teapot was constructed from six registered photographs; the gargoyle and dinosaur models were each built from nine images. *Reprinted from "Automatic Model Construction, Pose Estimation, and Object Recognition from Photographs Using Triangular Splines," by S. Sullivan and J. Ponce, IEEE Transactions on Pattern Analysis and Machine Intelligence, 20(10):1091–1096, (1998). © 1998 IEEE.*

to minimize the mean-squared ray-surface distance using a simple gradient descent technique. Although each distance is computed numerically, its derivatives with respect to the surface parameters V_{jk} are easily computed by differentiating the constraints satisfied by the surface and ray points where the distance is reached.

The three object models shown in Figure 26.5 have been constructed using the method described in this section. This technique does not require establishing any correspondence across the input pictures, but its scope is (currently) limited to static scenes. In contrast, the approach presented next is based on multicamera stereopsis, and, as such, requires correspondences, but it handles dynamic scenes as well as static ones.

Virtualized Reality Kanade and his colleagues (1997) have proposed the concept of *Virtualized Reality* as a new visual medium for manipulating and rendering prerecorded and synthetic images of real scenes captured in a controlled environment. The first physical implementation of this concept at Carnegie-Mellon University consisted of a geodesic dome equipped with 10 synchronized video cameras hooked to consumer-grade VCRs. As of this writing, the latest implementation is a "3D Room", where a volume of 20 × 20 × 9 cubic feet is observed by 49 color cameras connected to a PC cluster and registered in the same world coordinate system, with the capability of digitizing in real-time the synchronized video streams of all cameras. Three-dimensional scene models are acquired by fusing dense depth maps acquired via multiple-camera stereo (see Okutami and Kanade, 1993, chapter 11). One such map is acquired by each camera and a small number of its neighbors (between three and six). Every range image is then

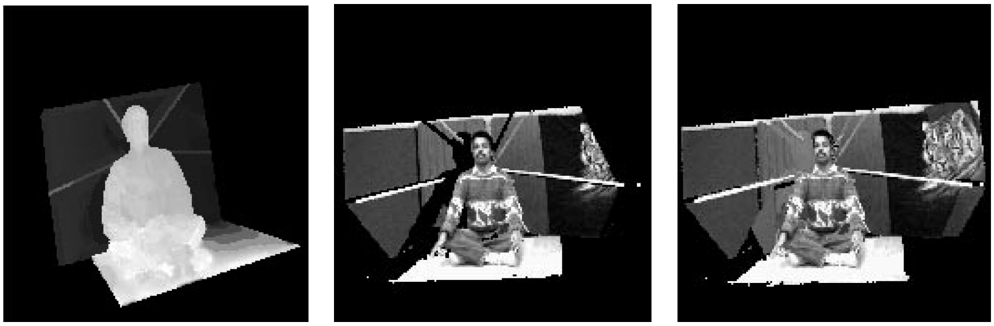

Figure 26.6 Multicamera stereo. From left to right: the range map associated with a cluster of cameras; a texture-mapped image of the corresponding mesh, observed from a different viewpoint; note the dark areas associated with depth discontinuities in the map; a texture-mapped image constructed from two adjacent camera clusters; note that the gaps have been filled. *Reprinted from "Virtualized Reality: Constructing Virtual Worlds From Real Scenes," by T. Kanade, P.W. Rander and J.P. Narayanan, IEEE Multimedia, 4(1):34–47, (1997). © 1997 IEEE.*

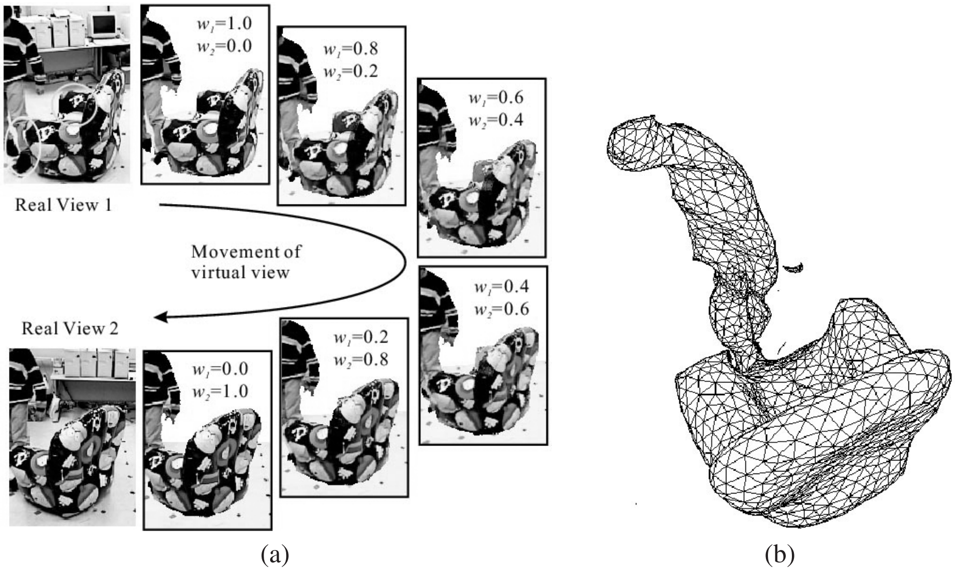

Figure 26.7 Virtualized Reality: (a) a sequence of synthetic images; note that occlusion in the two elliptical regions of the first view is handled correctly; (b) the corresponding mesh model. *Reprinted from "Appearance-Based Virtual View Generation of Temporally-Varying Events from Multi-Camera Images in the 3D Room," by H. Saito, S. Baba, M. Kimura, S. Vedula, and T. Kanade, Tech. Rep. CMU-CS-99-127, School of Computer Science, Carnegie-Mellon University, (1999).*

converted to a surface mesh that can be rendered using classical computer graphics techniques such as texture mapping. As shown by Figure 26.6, images of a scene constructed from a single depth map may exhibit gaps. These gaps can be filled by rendering in the same image the meshes corresponding to adjacent cameras.

It is also possible to directly merge the surface meshes associated with different cameras into a single surface model. This task is challenging since: (a) multiple, conflicting measurements of the same surface patches are available in areas where the fields of view of several cameras overlap, and (b) certain scene patches are not observed by any camera. Both problems can be solved using the volumetric technique for range image fusion proposed by Curless and Levoy (1996) and introduced in chapter 21. Once a global surface model has been constructed, it can of course be texture mapped as before. Synthetic animations can also be obtained by interpolating two arbitrary views in the input sequence. First, the surface model is used to establish correspondences between these two views: The optical ray passing through any point in the first image is intersected with the mesh and the intersection point is reprojected in the second image, yielding the desired match.[2] Once the correspondences are known, new views are constructed by linearly interpolating both the positions and colors of matching points. As discussed in Saito et al. (1999), this simple algorithm only provides an approximation of true perspective imaging, and additional logic has to be added in practice to handle points that are visible in the first image but not in the second one. Nevertheless, it can be used to generate realistic animations of dynamic scenes with changing occlusion patterns, as demonstrated by Figure 26.7.

26.1.2 Scene Modeling from Unregistered Images

This section addresses again the problem of acquiring and rendering three-dimensional object models from a set of images, but this time the positions of the cameras observing the scene are not known a priori and must be recovered from image information using methods related to those presented in chapters 12 and 13. The techniques presented in this section are, however, explicitly geared toward computer graphics applications.

The Façade System The *Façade* system for modeling and rendering architectural scenes from digitized photographs was developed at UC Berkeley by Debevec, Taylor, and Malik (1996). This system takes advantage of the relatively simple overall geometry of many buildings to simplify the estimation of scene structure and camera motion, and it uses the simple but powerful idea of *model-based stereopsis*, to be described in a minute, to add detail to rough building outlines. Figure 26.8 shows an example.

Façade models are constrained hierarchies of parametric primitives such as boxes, prisms, and solids of revolution. These primitives are defined by a small number of coefficients (e.g., the height, width, and breadth of a box) and related to each other by rigid transformations. Any of the parameters defining a model is either a constant or variable, and constraints can be specified between the various unknowns (e.g., two blocks may be constrained to have the same height). Model hierarchies are defined interactively with a graphical user interface, and the main computational task of the Façade system is to use image information to assign definite values to the unknown model parameters. The overall system is divided into three main components: The first one, or *photogrammetric module*, recasts structure and motion estimation as the nonlinear optimization problem of minimizing the discrepency between line segments selected by hand in the photographs and the projections of the corresponding parts of the parametric model (see Exercises for details). As shown in Debevec et al. (1996), this process involves relatively few

[2] Classical narrow-baseline methods like correlation would be ineffective in this context since the two views may be far from each other. A similar method is used in the Façade system described later in this chapter to establish correspondences between widely separated images when the rough shape of the observed surface is known.

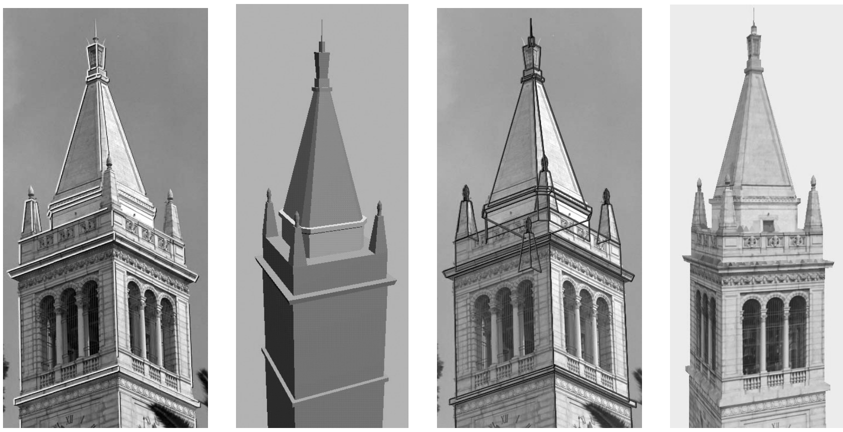

Figure 26.8 Façade model of the Berkeley Campanile. From left to right: A photograph of the Campanile, with selected edges overlaid; the 3D model recovered by photogrammetric modeling; reprojection of the model into the photograph; a texture-mapped view of the model. *Reprinted from "Modeling and Rendering Architecture from Photographs: A Hybrid Geometry- and Image-Based Approach," by P. Debevec, C.J. Taylor, and J. Malik, Proc. SIGGRAPH, (1996). © 1996 ACM, Inc. Included here by permission.*

variables, namely the positions and orientations of the cameras used to photograph a building and the parameters of the building model, and when the orientation of some of the model edges is fixed relative to the world coordinate system, an initial estimate for these parameters is easily found using linear least squares.

The second main component of Façade is the *view-dependent texture-mapping module* that renders an architectural scene by mapping different photographs onto its geometric model according to the user's viewpoint. Conceptually, the cameras are replaced by slide projectors that project the original images onto the model. Of course, each camera only sees a portion of a building, and several photographs must be used to render a complete model. In general, parts of a building are observed by several cameras, so the renderer must not only pick, but also appropriately merge, the pictures relevant to the synthesis of a virtual view. The solution adopted in Façade is to assign to each pixel in a new image a weighted average of the values predicted from the overlapping input pictures, with weights inversely proportional to the angle between the corresponding light rays in the input and virtual views.

The last component of Façade is the *model-based stereopsis module*, which uses stereo pairs to add fine geometric detail to the relatively rough scene description constructed by the photogrammetric modeling module. The main difficulty in using stereo vision in this setting is the wide separation of the cameras, which prevents the straightforward use of correlation-based matching techniques. The solution adopted in Façade is to exploit a priori shape information to map the stereo images into the same reference frame (Figure 26.9, top). Specifically, given *key* and *offset* pictures, the offset image can be projected onto the scene model before being rendered from the key camera's viewpoint, yielding a *warped offset* picture similar to the key image (Figure 26.9, bottom). In turn, this allows the use of correlation to establish correspondences between these two images, and thus between the key and offset images as well. Once the matches between these two pictures have been established, stereo reconstruction reduces to the usual triangulation process.

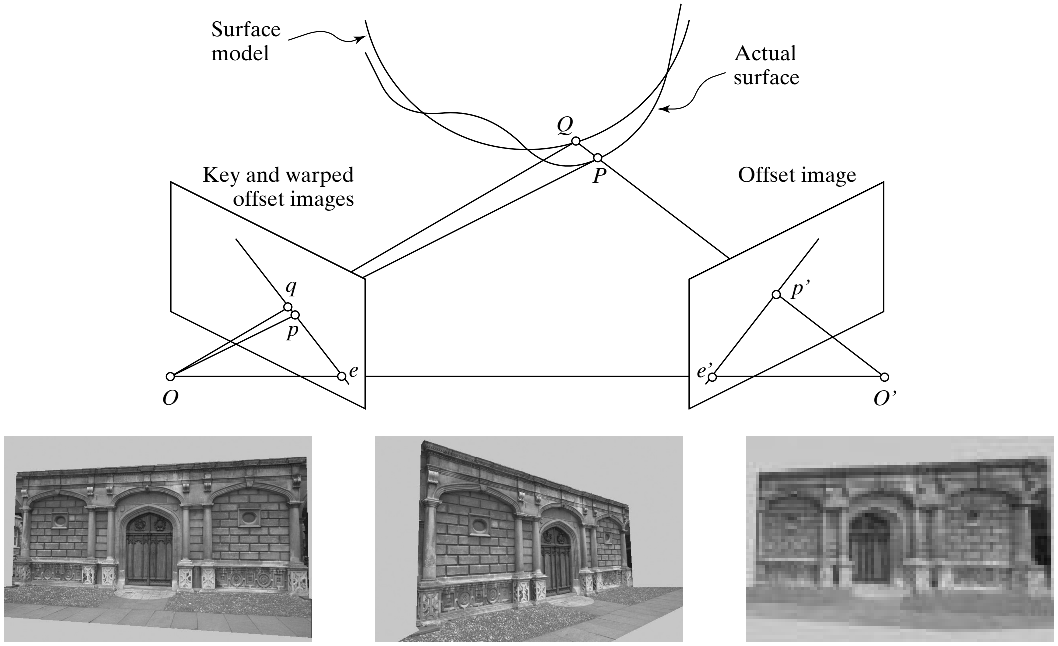

Figure 26.9 Model-based stereopsis. Top: Synthesis of a warped offset image. The point p' in the offset image is mapped onto the point Q of the surface model, then reprojected onto the point q of the warped offset image. The actual surface point P observed by both cameras projects onto the point p of the key image. Note that the point q must lie on the epipolar line ep, which facilitates the search for matches as in the conventional stereo case. Note also that the disparity between p and q along the epipolar line measures the discrepancy between the modeled and actual surfaces. After Debevec et al. (1996, Figure 15). Bottom, from left to right: A key image, an offset image, and the corresponding warped offset image. *Reprinted from "Modeling and Rendering Architecture from Photographs: A Hybrid Geometry- and Image-Based Approach," by P. Debevec, C.J. Taylor, and J. Malik, Proc. SIGGRAPH, (1996). © 1996 ACM, Inc. Included here by permission.*

26.2 TRANSFER-BASED APPROACHES TO IMAGE-BASED RENDERING

This section explores a completely different approach to image-based rendering. In this framework, an explicit three-dimensional scene reconstruction is never performed. Instead, new images are created directly from a (possibly small) set of views among which point correspondences have been established by feature tracking or conventional stereo matching. This approach is related to the classical transfer problem from photogrammetry already mentioned in chapter 10. Given the image positions of a number of *tie points* in a set of reference images and in a new image, and given the image positions of a ground point in the reference images, predict the position of that point in the new image.

Transfer-based techniques for image-based rendering were introduced in the projective setting by Laveau and Faugeras (1994), who proposed to first estimate the pairwise epipolar geometry between reference views, then reproject the scene points into a virtual image, specified by the projections of the new optical center in two reference pictures (i.e., the epipoles) and the position of four tie points in the new view. By definition, the epipolar geometry constrains the

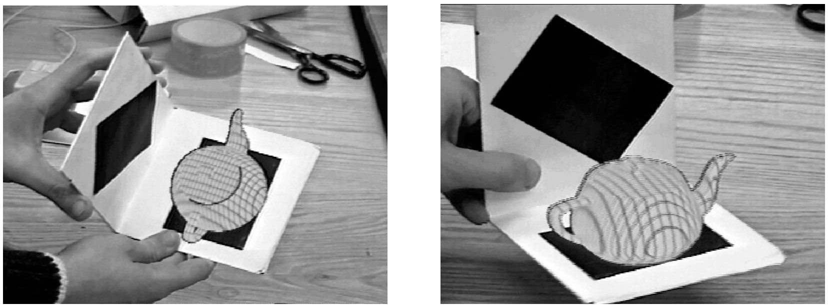

Figure 26.10 Augmented reality experiment. The (affine) world coordinate system is defined by corners of the black polygons. *Reprinted from "Calibration-Free Augmented Reality," by K. Kutulakos and J. Vallino, IEEE Transactions on Visualization and Computer Graphics, 4(1):1–20, (1998). © 1998 IEEE.*

possible reprojections of points in the reference images. In the new view, the projection of the scene point is at the intersection of the two epipolar lines associated with the point and two reference pictures. Once the feature points have been reprojected, realistic pictures are synthesized using ray tracing and texture mapping. As noted by Laveau and Faugeras, however, since the Euclidean constraints associated with calibrated cameras are not enforced, the rendered images are separated from correct pictures by arbitrary planar projective transformations unless additional scene constraints are taken into account. The rest of this section explores two affine variants of the transfer-based approach that circumvent this difficulty. Both techniques construct a parameterization of the set of all images of a rigid scene: In the first case (Section 26.2.1), the affine structure of the space of affine images is used to render synthetic objects in an augmented reality system. Because the tie points in this case are always geometrically valid image features (e.g., the corners of calibration polygons; see Figure 26.10), the synthesized images are automatically Euclidean ones. In the second instance (Section 26.2.2), the metric constraints associated with calibrated cameras are explicitly taken into account in the image space parameterization, guaranteeing once again the synthesis of correct Euclidean images.

Let us note again a particularity of transfer-based approaches to image-based rendering already mentioned in the introduction: Because no three-dimensional model is ever constructed, a joystick cannot be used to control the synthesis of an animation. Instead, the position of tie points must be specified interactively by a user. This is not a problem in an augmented reality context, but whether this is a viable user interface for virtual reality applications remains to be shown.

26.2.1 Affine View Synthesis

Here we address the problem of synthesizing new (affine) images of a scene from old ones *without* setting an explicit three-dimensional Euclidean coordinate system. Recall from chapter 12 that if we denote the coordinate vector of a scene point P in some world coordinate system by $\boldsymbol{P} = (x, y, z)^T$ and denote by $\boldsymbol{p} = (u, v)^T$ the coordinate vector of the projection p of P onto the image plane, the affine camera model of Eq. (2.19) can be written as

$$\boldsymbol{p} = \mathcal{A}\boldsymbol{P} + \boldsymbol{b}, \quad \text{where} \quad \mathcal{A} = \begin{pmatrix} \boldsymbol{a}_1^T \\ \boldsymbol{a}_2^T \end{pmatrix}, \tag{26.1}$$

$\boldsymbol{b}$ is the position of the projection into the image of the object coordinate system's origin, and $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$ are vectors in $\mathbb{R}^3$.

Let us consider four (noncoplanar) scene points, say P_0, P_1, P_2, and P_3. We can choose (without loss of generality) these points as an affine reference frame so their coordinate vectors are

$$\boldsymbol{P}_0 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \quad \boldsymbol{P}_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \quad \boldsymbol{P}_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \quad \boldsymbol{P}_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}.$$

The points P_i $(i = 1, 2, 3)$ are *not* in general at a unit distance from P_0, nor are the vectors $\overrightarrow{P_0P_i}$ and $\overrightarrow{P_0P_j}$ orthogonal to each other when $i \neq j$. This is irrelevant since we work in an affine setting. Since the 3×3 matrix with columns $\boldsymbol{P}_1$, $\boldsymbol{P}_2$, and $\boldsymbol{P}_3$ is the identity, Eq. (26.1) can be rewritten as

$$\boldsymbol{p} = \mathcal{A}\boldsymbol{P} + \boldsymbol{b} = \begin{pmatrix} \boldsymbol{a}_1^T \\ \boldsymbol{a}_2^T \end{pmatrix} [\boldsymbol{P}_1 | \boldsymbol{P}_2 | \boldsymbol{P}_3] \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \boldsymbol{b}.$$

Finally, since we have chosen P_0 as the origin of the world coordinate system, we have $\boldsymbol{b} = \boldsymbol{p}_0$ and we obtain

$$\boldsymbol{p} = (1 - x - y - z)\boldsymbol{p}_0 + x\boldsymbol{p}_1 + y\boldsymbol{p}_2 + z\boldsymbol{p}_3. \tag{26.2}$$

This result is related to the affine structure of affine images as discussed in chapter 12. In the context of image-based rendering, it follows from Eq. (26.2) that x, y, and z can be computed from $m \geq 2$ images of the points P_0, P_1, P_2, P_3, and P through linear least squares. Once these values are known, new images can be synthesized by specifying the image positions of the points p_0, p_1, p_2, p_3 and using Eq. (26.2) to compute all the other point positions (Kutulakos and Vallino, 1998). In addition, since the affine representation of the scene is truly three-dimensional, the relative depth of scene points can be computed and used to eliminate hidden surfaces in the z-buffer part of the graphics pipeline. It should be noted that specifying arbitrary positions for the points p_0, p_1, p_2, p_3 generally produces affinely deformed pictures. This is not a problem in augmented reality applications, where graphical and physical objects co-exist in the image. In this case, the anchor points p_0, p_1, p_2, p_3 can be chosen among true image points, guaranteed to be in the correct Euclidean position. Figure 26.10 shows an example where synthetic objects have been overlaid on real images.

When longer image sequences are available, a variant of this approach that takes into account all scene points in a uniform manner can be obtained as follows. Suppose we observe a fixed set of points $P_0, \dots, P_{n-1}$ with coordinate vectors $\boldsymbol{P}_i$ $(i = 0, \dots, n-1)$ and let $\boldsymbol{p}_i$ denote the coordinate vectors of the corresponding image points. Writing Eq. (26.1) for all the scene points yields

$$\begin{pmatrix} \boldsymbol{p}_0 \\ \dots \\ \boldsymbol{p}_{n-1} \end{pmatrix} = \begin{pmatrix} \boldsymbol{P}_0^T & \boldsymbol{0}^T & 1 & 0 \\ \boldsymbol{0}^T & \boldsymbol{P}_0^T & 0 & 1 \\ \dots & \dots & \dots & \dots \\ \boldsymbol{P}_{n-1}^T & \boldsymbol{0}^T & 1 & 0 \\ \boldsymbol{0}^T & \boldsymbol{P}_{n-1}^T & 0 & 1 \end{pmatrix} \begin{pmatrix} \boldsymbol{a}_1 \\ \boldsymbol{a}_2 \\ \boldsymbol{b} \end{pmatrix}.$$

In other words, the set of all affine images of *n fixed* points is an eight-dimensional vector space V embedded in $\mathbb{R}^{2n}$ and parameterized by the vectors $\boldsymbol{a}_1$, $\boldsymbol{a}_2$, and $\boldsymbol{b}$.[3] Given $m \geq 8$ views

[3] This does not contradict the result established in chapter 12, which states that the set of *m fixed* views of an *arbitrary* collection of points is a three-dimensional affine subspace of $\mathbb{R}^{2m}$.

of $n \geq 4$ points, a basis for this vector space can be identified by performing the singular value decomposition of the $2n \times m$ matrix

$$\begin{pmatrix} \boldsymbol{p}_0^{(1)} & \dots & \boldsymbol{p}_0^{(m)} \\ \dots & \dots & \dots \\ \boldsymbol{p}_{n-1}^{(1)} & \dots & \boldsymbol{p}_{n-1}^{(m)} \end{pmatrix},$$

where $\boldsymbol{p}_i^{(j)}$ denotes the position of the image point number i in frame number j.[4] Once a basis for V has been constructed, new images can be constructed by assigning arbitrary values to $\boldsymbol{a}_1$, $\boldsymbol{a}_2$ and $\boldsymbol{b}$. For interactive image synthesis purposes, a more intuitive control of the imaging geometry can be obtained by specifying as before the position of four image points, solving for the corresponding values of $\boldsymbol{a}_1$, $\boldsymbol{a}_2$, and $\boldsymbol{b}$, and computing the remaining image positions.

26.2.2 Euclidean View Synthesis

As discussed earlier, a drawback of the method presented in the previous section is that specifying arbitrary positions for the points p_0, p_1, p_2, p_3 generally yields affinely deformed pictures. This can be avoided by taking into account from the start the Euclidean constraints associated with calibrated cameras. We saw in chapter 12 that a weak-perspective camera is an affine camera satisfying the two quadratic constraints

$$\boldsymbol{a}_1 \cdot \boldsymbol{a}_2 = 0 \quad \text{and} \quad |\boldsymbol{a}_1|^2 = |\boldsymbol{a}_2|^2.$$

The previous section showed that the affine images of a fixed scene form an eight-dimensional vector space V. Now if we restrict our attention to weak-perspective cameras, the set of images becomes the six-dimensional subspace defined by these two polynomial constraints. Similar constraints apply to paraperspective and true perspective projection, and they also define a six-dimensional *variety* (i.e., a subspace defined by polynomial equations) in each case.

Let us suppose that we observe three points P_0, P_1, P_2 whose images are not collinear. We can choose (without loss of generality) a *Euclidean* coordinate system such that the coordinate vectors of the four points in this system are

$$\boldsymbol{P}_0 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \quad \boldsymbol{P}_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \quad \boldsymbol{P}_2 = \begin{pmatrix} p \\ q \\ 0 \end{pmatrix},$$

where p and q are nonzero, but (a priori) unknown. Let us denote as before by p_i the projection of the point P_i ($i = 0, 1, 2$). Since P_0 is the origin of the world coordinate system, we have $\boldsymbol{b} = \boldsymbol{p}_0$. We are also free to pick p_0 as the origin of the image coordinate system (this amounts to submitting all image points to a known translation), so Eq. (26.1) simplifies into

$$\boldsymbol{p} = \mathcal{A}\boldsymbol{P} = \begin{pmatrix} \boldsymbol{a}_1^T \boldsymbol{P} \\ \boldsymbol{a}_2^T \boldsymbol{P} \end{pmatrix}. \tag{26.3}$$

Now applying Eq. (26.3) to P_1, P_2, and P yields

$$\boldsymbol{u} \overset{\text{def}}{=} \begin{pmatrix} u_1 \\ u_2 \\ u \end{pmatrix} = \mathcal{P}\boldsymbol{a}_1 \quad \text{and} \quad \boldsymbol{v} \overset{\text{def}}{=} \begin{pmatrix} v_1 \\ v_2 \\ v \end{pmatrix} = \mathcal{P}\boldsymbol{a}_2, \tag{26.4}$$

[4] Requiring at least eight images may seem like overkill since the affine structure of a scene can be recovered from two pictures as shown in chapter 12. Indeed, as shown in the exercises, a basis for V can in fact be constructed from two images of at least four points.

where

$$\mathcal{P} \overset{\text{def}}{=} \begin{pmatrix} \boldsymbol{P}_1^T \\ \boldsymbol{P}_2^T \\ \boldsymbol{P}^T \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ p & q & 0 \\ x & y & z \end{pmatrix}.$$

In turn, this implies that

$$\boldsymbol{a}_1 = \mathcal{Q}\boldsymbol{u} \quad \text{and} \quad \boldsymbol{a}_2 = \mathcal{Q}\boldsymbol{v}, \tag{26.5}$$

where

$$\mathcal{Q} \overset{\text{def}}{=} \mathcal{P}^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ \lambda & \mu & 0 \\ \alpha/z & \beta/z & 1/z \end{pmatrix} \quad \text{and} \quad \begin{cases} \lambda = -p/q, \\ \mu = 1/q, \\ \alpha = -(x + \lambda y), \\ \beta = -\mu y. \end{cases}$$

Using Eq. (26.5) and letting $\mathcal{R} \overset{\text{def}}{=} z^2 \mathcal{Q}^T \mathcal{Q}$, the weak-perspective constraints of Eq. (12.10) can be rewritten as

$$\begin{cases} \boldsymbol{u}^T \mathcal{R} \boldsymbol{u} - \boldsymbol{v}^T \mathcal{R} \boldsymbol{v} = 0, \\ \boldsymbol{u}^T \mathcal{R} \boldsymbol{v} = 0, \end{cases} \tag{26.6}$$

with

$$\mathcal{R} = \begin{pmatrix} \xi_1 & \xi_2 & \alpha \\ \xi_2 & \xi_3 & \beta \\ \alpha & \beta & 1 \end{pmatrix} \quad \text{and} \quad \begin{cases} \xi_1 = (1 + \lambda^2) z^2 + \alpha^2, \\ \xi_2 = \lambda \mu z^2 + \alpha\beta, \\ \xi_3 = \mu^2 z^2 + \beta^2. \end{cases}$$

Equation (26.6) defines a pair of linear constraints on the coefficients ξ_i $(i = 1, 2, 3)$, α, and β. These can be rewritten as

$$\begin{pmatrix} \boldsymbol{d}_1^T \\ \boldsymbol{d}_2^T \end{pmatrix} \boldsymbol{\xi} = 0, \tag{26.7}$$

where

$$\boldsymbol{d}_1 \overset{\text{def}}{=} \begin{pmatrix} u_1^2 - v_1^2 \\ 2(u_1 u_2 - v_1 v_2) \\ u_2^2 - v_2^2 \\ 2(u_1 u - v_1 v) \\ 2(u_2 u - v_2 v) \\ u^2 - v^2 \end{pmatrix}, \quad \boldsymbol{d}_2 \overset{\text{def}}{=} \begin{pmatrix} u_1 v_1 \\ u_1 v_2 + u_2 v_1 \\ u_2 v_2 \\ u_1 v + u v_1 \\ u_2 v + u v_2 \\ uv \end{pmatrix}, \quad \text{and} \quad \boldsymbol{\xi} \overset{\text{def}}{=} \begin{pmatrix} \xi_1 \\ \xi_2 \\ \xi_3 \\ \alpha \\ \beta \\ 1 \end{pmatrix}.$$

When the four points P_0, P_1, P_2, and P are rigidly attached to each other, the five structure coefficients ξ_1, ξ_2, ξ_3, α, and β are fixed. For a rigid scene formed by n points, choosing three of the points as a reference triangle and writing Eq. (26.7) for the remaining ones yields a set of $2n - 6$ quadratic equations in $2n$ unknowns, which define a parameterization of the set of all weak-perspective images of the scenes. This is the *parameterized image variety (PIV)* of Genc and Ponce (2001).

Note again that the weak-perspective constraints of Eq. (26.7) are linear in the five structure coefficients. Thus, given a collection of images and point correspondences, these coefficients can be estimated through linear least squares. Once the vector $\boldsymbol{\xi}$ has been estimated, arbitrary image

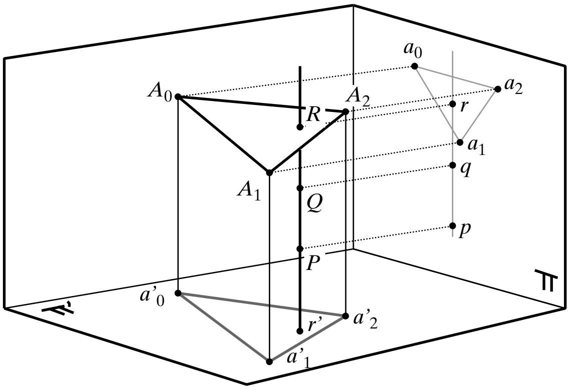

Figure 26.11 Z-buffering. *Reprinted from "Parameterized Image Varieties: A Novel Approach to the Analysis and Synthesis of Image Sequences," by Y. Genc and J. Ponce, Proc. International Conference on Computer Vision, (1998). © 1998 IEEE.*

positions can be assigned to the three reference points. Equation (26.7) yields, for each feature point, two quadratic constraints on the two unknowns u and v. Although this system should *a priori* admit four solutions, it admits, as shown in the exercises, exactly two real solutions. In fact, given n point correspondences and the image positions of the three tie points, it can also be shown (Genc and Ponce, 2001) that the pictures of the remaining $n - 3$ points can be determined in closed form up to a two-fold ambiguity. Once the positions of all feature points have been determined, the scene can be rendered by triangulating these points and texture-mapping the triangles. Interestingly, hidden-surface removal can also be performed via traditional z-buffer techniques, although no explicit three-dimensional reconstruction is performed: The idea is to assign relative depth values to the vertices of the triangulation, and it is closely related to the method used in the affine structure-from-motion theorem from chapter 12. Let Π denote the image plane of one of our input images, and Π' the image plane of our synthetic image. To render correctly two points P and Q that project onto the same point r' in the synthetic image, we must compare their depths (Figure 26.11).

Let R denote the intersection of the viewing ray joining P to Q with the plane spanned by the reference points A_0, A_1, and A_2, and let p, q, r denote the projections of P, Q, and R into the reference image. Suppose for the time being that P and Q are two of the points tracked in the input image; it follows that the positions of p and q are known. The position of r is easily computed by remarking that its coordinates in the affine basis of Π formed by the projections a_0, a_1, a_2 of the reference points are the same as the coordinates of R in the affine basis formed by the points A_0, A_1, A_2 in their own plane, and thus are also the same as the coordinates of r' in the affine basis of Π' formed by the projections a'_0, a'_1, a'_2 of the reference points. The ratio of the depths of P and Q relative to the plane Π is simply the ratio $\overline{pr}/\overline{qr}$. Not that deciding which point is actually visible requires orienting the line supporting the points p, q, r, which is simply the epipolar line associated with the point r'. A coherent orientation should be chosen for all epipolar lines (this is easy since they are all parallel to each other). Note that this does not require explicitly computing the epipolar geometry: Given a first point p', one can orient the line pr and then use the same orientation for all other point correspondences. The orientations chosen should also be consistent over successive frames, but this is not a problem since the direction of the epipolar lines changes slowly from one frame to the next,

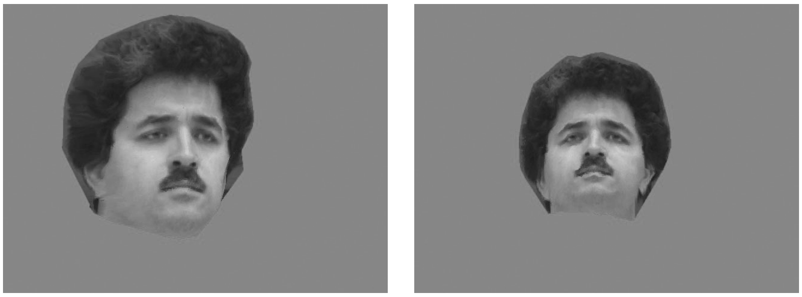

Figure 26.12 Two images of a face synthesized using parameterized image varieties. *Courtesy of Yakup Genc.*

and one can simply choose the new orientation so that it makes an acute angle with the previous one. Examples of synthetic pictures constructed using this method are shown in Figure 26.12.

26.3 THE LIGHT FIELD

This section discusses a different approach to image-based rendering, whose only similarity with the techniques discussed in the previous section is that, like them, it does not require the construction of any implicit or explicit 3D model of a scene. Let us consider, for example, a panoramic camera that optically records the radiance along rays passing through a single point and covering a full hemisphere (see, e.g., Peri and Nayar, 1997; Figure 26.13, left). It is possible to create

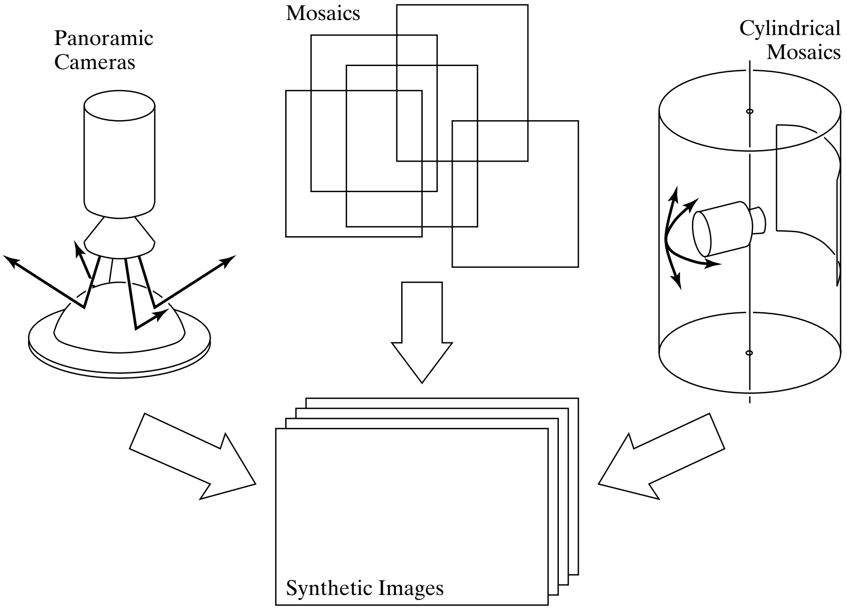

Figure 26.13 Constructing synthetic views of a scene from a fixed viewpoint.

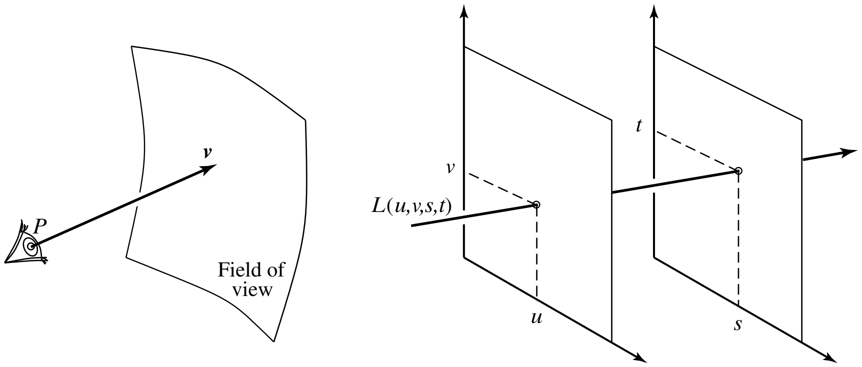

Figure 26.14 The plenoptic function and the light field. Left: the plenoptic function can be parameterized by the position P of the observer and the viewing direction $\boldsymbol{v}$. Right: the light field can be parameterized by the four parameters u, v, s, t defining a light slab. In practice, several light slabs are necessary to model a whole object and obtain full spherical coverage.

any image observed by a virtual camera whose pinhole is located at this point by mapping the original image rays onto virtual ones. This allows a user to arbitrarily pan and tilt the virtual camera and interactively explore his or her visual environment. Similar effects can be obtained by stitching together close-by images taken by a hand-held camcorder into a mosaic (see, e.g., Shum and Szeliski, 1998; Figure 26.13, middle), or by combining the pictures taken by a camera panning (and possibly tilting) about its optical center into a cylindrical mosaic (see, e.g., Chen, 1995; Figure 26.13, right).

These techniques have the drawback of limiting the viewer motions to pure rotations about the optical center of the camera. A more powerful approach can be devised by considering the *plenoptic function* (Adelson and Bergen, 1991) that associates with each point in space the (wavelength-dependent) radiant energy along a ray passing through this point at a given time (Figure 26.14, left). The *light field* (Levoy and Hanrahan, 1996) is a snapshot of the plenoptic function for light traveling in vacuum in the absence of obstacles. This relaxes the dependence of the radiance on time and on the position of the point of interest along the corresponding ray (since radiance is constant along straight lines in a nonabsorbing medium) and yields a representation of the plenoptic function by the radiance along the four-dimensional set of light rays. These rays can be parameterized in many different ways (e.g., using the Plücker coordinates introduced in chapter 3), but a convenient parameterization in the context of image-based rendering is the *light slab*, where each ray is specified by the coordinates of its intersections with two arbitrary planes (Figure 26.14, right).

The light slab is the basis for a two-stage approach to image-based rendering. During the learning stage, many views of a scene are used to create a discrete version of the slab that can be thought of as a four-dimensional lookup table. At synthesis time, a virtual camera is defined, and the corresponding view is interpolated from the lookup table. The quality of the synthesized images depends on the number of reference images. The closer the virtual view is to the reference images, the better the quality of the synthesized image. Note that constructing the light slab model of the light field does not require establishing correspondences between images. It should be noted that, unlike most methods for image-based rendering that rely on texture mapping and thus assume (implicitly) that the observed surfaces are Lambertian, light-field techniques can be used to render (under a fixed illumination) pictures of objects with *arbitrary* BRDFs.

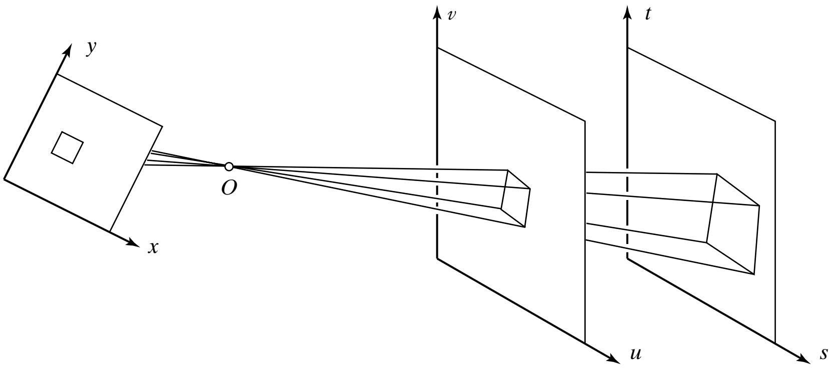

Figure 26.15 The acquisition of a light slab from images and the synthesis of new images from a light slab can be modeled via projective transformations between the (x, y) image plane and the (u, v) and (s, t) planes defining the slab.

In practice, a sample of the light field is acquired by taking a large number of images and mapping pixel coordinates onto slab coordinates. Figure 26.15 illustrates the general case: The mapping between any pixel in the (x, y) image plane and the corresponding areas of the (u, v) and (s, t) plane defining a light slab is a planar projective transformation. Hardware- or software-based texture mapping can thus be used to populate the light field on a four-dimensional rectangular grid. In the experiments described in Levoy and Hanrahan (1996), light slabs are acquired in the simple setting of a camera mounted on a planar gantry and equipped with a pan-tilt head so it can rotate about its optical center and always point toward the center of the object of interest. In this context, all calculations can be simplified by taking the (u, v) plane to be the plane in which the camera's optical center is constrained to remain.

At rendering time, the projective mapping between the (virtual) image plane and the two planes defining the light slab can once again be used to efficiently synthesize new images. Figure 26.16 shows sample pictures generated using the light-field approach. The top three image pairs were generated using synthetic pictures of various objects to populate the light field. The last pair of images was constructed by using the planar gantry mentioned earlier to acquire 2048 256 × 256 images of a toy lion, grouped into four slabs each consisting of 32 × 16 images.

An important issue is the size of the light slab representation: The raw input images of the lion take 402MB of disk space. There is, of course, much redundancy in these pictures, as in the case of successive frames in a motion sequence. A simple but effective two-level approach to image (de)compression is proposed in Levoy and Hanrahan (1996): The light slab is first decomposed into four-dimensional tiles of color values. These tiles are encoded using *vector quantization* (Gersho and Gray, 1992), a lossy compression technique where the 48-dimensional vectors representing the RGB values at the 16 corners of the original tiles are replaced by a relatively small set of reproduction vectors, called *codewords*, that best approximate in the mean-squared-error sense the input vectors. The light slab is thus represented by a set of indexes in the *codebook* formed by all codewords. In the case of the lion, the codebook is relatively small (0.8MB) and the size of the set of indexes is 16.8MB. The second compression stage consists of applying the *gzip* implementation of *entropy coding* (Ziv and Lempel, 1977) to the codebook and the indexes. The final size of the representation is only 3.4MB, corresponding to a compression rate of 118:1. At rendering time, entropy decoding is performed as the file is loaded in main memory. Dequantization is performed on demand during display, and it allows interactive refresh rates.

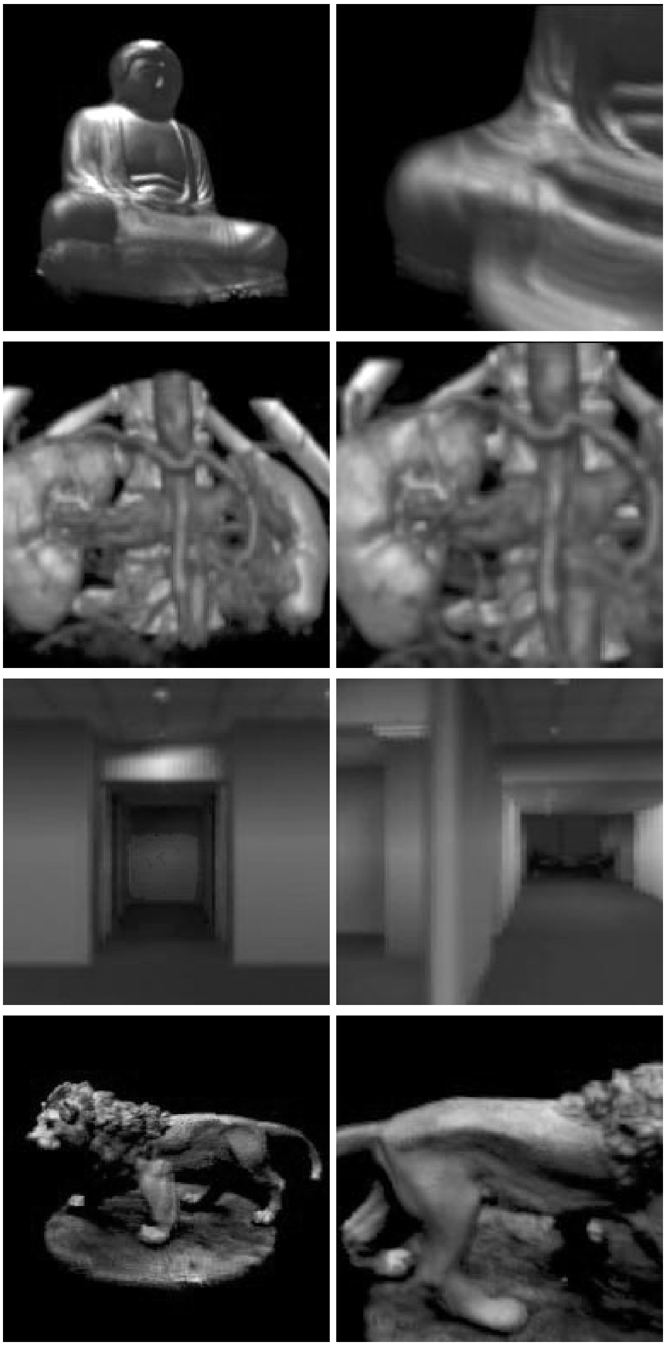

Figure 26.16 Images synthesized with the light field approach. *Reprinted from "Light Field Rendering," by M. Levoy and P. Hanrahan, Proc. SIGGRAPH, (1996). © 1996 ACM, Inc. Included here by permission.*

26.4 NOTES

Image-based rendering is a quickly expanding field. To close this chapter, let us just mention a few alternatives to the approaches already mentioned in the previous sections. The intersection of all of the cones that graze the surface of a solid forms its *visual hull* (Laurentini, 1995). A solid is always contained in its visual hull, which, in turn, is contained in the solid's convex hull. The volumetric approach to object modeling from registered silhouettes presented in Section 26.1 is aimed at constructing an approximation of the visual hull from a finite set of photographs. Variants use polyhedra or octrees (Martin and Aggarwal, 1983, Connolly and Stenstrom, 1989, Srivastava and Ahuja, 1990) to represent the cones and their intersection, and include a commercial system, Sphinx3D (Niem and Buschmann, 1994), for automatically constructing polyhedral models from images. See also Kutulakos and Seitz (1999) for a related approach, called *space carving*, where empty voxels are iteratively removed using brightness or color coherence constraints. The tangency constraint has been used in various approaches for reconstructing a surface from a continuous sequence of outlines under known or unknown camera motions (Arbogast and Mohr, 1991, Cipolla and Blake, 1992, Vaillant and Faugeras, 1992, Cipolla et al., 1995, Boyer and Berger, 1996, Cross et al., 1999, Joshi et al., 1999). Variants of the view interpolation method discussed in Section 26.1 include Williams and Chen (1993) and Seitz and Dyer (1995, 1996). Transfer-based approaches to image-based rendering include, besides those discussed in Section 26.2, Havaldar et al. (1996) and Avidan and Shashua (1997). As briefly mentioned in Section 26.3, a number of techniques have been developed for interactively exploring a user's visual environment from a fixed viewpoint. These include a commercial system, *QuickTime VR*, developed at Apple by Chen (1995), and algorithms that reconstruct pinhole perspective images from panoramic pictures acquired by special-purpose cameras (see, e.g., Peri and Nayar, 1997). Similar effects can be obtained in a less controlled setting by stitching together close-by images taken by a hand-held camcorder into a mosaic (see, e.g., Irani et al., 1996, Shum and Szeliski, 1998). For images of distant terrains or cameras rotating about their optical center, the mosaic can be constructed by registering successive pictures via planar homographies. In this context, estimating the *optical flow* (i.e., the vector field of apparent image velocities at every image point, a notion that has, admittedly, largely been ignored in this book), may also prove important for fine registration and *deghosting* (Shum and Szeliski, 1998). Variants of the light field approach discussed in Section 26.3 include McMillan and Bishop (1995) and Gortler et al. (1996). An excellent introduction to Bézier arcs and patches and spline curves and surfaces can be found in Farin (1993).

PROBLEMS

26.1. Given $n+1$ point $P_0, \dots, P_n$, we recursively define the parametric curve $P_i^k(t)$ by $P_i^0(t) = P_i$ and

$$P_i^k(t) = (1-t)P_i^{k-1}(t) + tP_{i+1}^{k-1}(t) \quad \text{for} \quad k = 1, \dots, n \quad \text{and} \quad i = 0, \dots, n-k.$$

We show in this exercise that $P_0^n(t)$ is the Bézier curve of degree n associated with the $n+1$ points $P_0, \dots, P_n$. This construction of a Bézier curve is called the *de Casteljeau algorithm*.

(a) Show that Bernstein polynomials satisfy the recursion

$$b_i^{(n)}(t) = (1-t)b_i^{(n-1)}(t) + tb_{i-1}^{(n-1)}(t)$$

with $b_0^{(0)}(t) = 1$ and, by convention, $b_j^{(n)}(t) = 0$ when $j < 0$ or $j > n$.

(b) Use induction to show that

$$P_i^k(t) = \sum_{j=0}^{k} b_j^{(k)}(t) P_{i+j} \quad \text{for} \quad k = 0, \dots, n \quad \text{and} \quad i = 0, \dots, n-k.$$

26.2. Consider a Bézier curve of degree n defined by $n+1$ control points $P_0, \dots, P_n$. We address here the problem of constructing the $n+2$ control points $Q_0, \dots, Q_{n+1}$ of a Bézier curve of degree $n+1$ with the same shape. This process is called *degree elevation*. Show that $Q_0 = P_0$ and

$$Q_j = \frac{j}{n+1} P_{j-1} + \left(1 - \frac{j}{n+1}\right) P_j \quad \text{for} \quad j = 1, \dots, n+1.$$

Hint: Write that the same point is defined by the barycentric combinations associated with the two curves, and equate the polynomial coefficients on both sides of the equation.

26.3. Show that the tangent to the Bézier curve $P(t)$ defined by the $n+1$ control points $P_0, \dots, P_n$ is

$$P'(t) = n \sum_{j=0}^{n-1} b_j^{(n-1)}(t)(P_{j+1} - P_j).$$

Conclude that the tangents at the endpoints of a Bézier arc are along the first and last line segments of its control polygon.

26.4. Show that the construction of the points Q_i in Section 26.1.1 places these points in a plane that passes through the centroid O of the points C_i.

26.5. Façade's photogrammetric module. We saw in the exercises of Chapter 3 that the mapping between a line δ with Plücker coordinate vector $\boldsymbol{\Delta}$ and its image δ with homogeneous coordinates $\boldsymbol{\Delta}$ can be represented by $\rho\boldsymbol{\delta} = \tilde{\mathcal{M}}\boldsymbol{\Delta}$. Here, $\boldsymbol{\Delta}$ is a function of the model parameters, and $\tilde{\mathcal{M}}$ depends on the corresponding camera position and orientation.

(a) Assuming that the line δ has been matched with an image edge e of length l, a convenient measure of the discrepancy between predicted and observed data is obtained by multiplying by l the mean squared distance separating the points of e from δ. Defining $d(t)$ as the signed distance between the edge point $p = (1-t)p_0 + tp_1$ and the line δ, show that

$$E = \int_0^1 d^2(t)\, dt = \frac{1}{3}\left(d(0)^2 + d(0)d(1) + d(1)^2\right),$$

where d_0 and d_1 denote the (signed) distances between the endpoints of e and δ.

(b) If $\boldsymbol{p}_0$ and $\boldsymbol{p}_1$ denote the homogeneous coordinate vectors of these points, show that

$$d_0 = \frac{1}{|[\tilde{\mathcal{M}}\boldsymbol{\Delta}]_2|} \boldsymbol{p}_0^T \tilde{\mathcal{M}}\boldsymbol{\Delta} \quad \text{and} \quad d_1 = \frac{1}{|[\tilde{\mathcal{M}}\boldsymbol{\Delta}]_2|} \boldsymbol{p}_1^T \tilde{\mathcal{M}}\boldsymbol{\Delta},$$

where $[\boldsymbol{a}]_2$ denotes the vector formed by the first two coordinates of the vector $\boldsymbol{a}$ in $\mathbb{R}^3$.

(c) Formulate the recovery of the camera and model parameters as a non-linear least-squares problem.

26.6. Show that a basis for the eight-dimensional vector space V formed by all affine images of a fixed set of points $P_0, \dots, P_{n-1}$ can be constructed from at least two images of these points when $n \geq 4$.

Hint: Use the matrix

$$\begin{pmatrix} u_0^{(1)} & v_0^{(1)} & \dots & u_0^{(m)} & v_0^{(m)} \\ \dots & \dots & \dots & & \\ u_{n-1}^{(1)} & v_{n-1}^{(1)} & \dots & u_{n-1}^{(m)} & v_{n-1}^{(m)} \end{pmatrix},$$

where $(u_i^{(j)}, v_i^{(j)})$ are the coordinates of the projection of the point P_i into image number j.

26.7. Show that the set of all projective images of a fixed scenes is an eleven-dimensional variety.

26.8. Show that the set of all perspective images of a fixed scene (for a camera with constant intrinsic parameters) is a six-dimensional variety.

26.9. In this exercise, we show that Eq. (26.7) only admits two solutions.

(a) Show that Eq. (26.6) can be rewritten as

$$\begin{cases} X^2 - Y^2 + e_1 - e_2 = 0, \\ 2XY + e = 0, \end{cases} \tag{26.8}$$

where

$$\begin{cases} X = u + \alpha u_1 + \beta u_2, \\ Y = v + \alpha v_1 + \beta v_2, \end{cases}$$

and e, e_1, and e_2 are coefficients depending on u_1, v_1, u_2, v_2 and the structure parameters.

(b) Show that the solutions of Eq. (26.8) are given by

$$\begin{cases} X' = \sqrt[4]{(e_1 - e_2)^2 + e^2} \cos\left(\frac{1}{2} \arctan(e, e_1 - e_2)\right), \\ Y' = \sqrt[4]{(e_1 - e_2)^2 + e^2} \sin\left(\frac{1}{2} \arctan(e, e_1 - e_2)\right), \end{cases}$$

and $(X'', Y'') = (-X', -Y')$.

Hint: Use a change of variables to rewrite Eq. (26.8) as a system of trigonometric equations.