Interactive 3D Modeling from Multiple Images using Scene Regularities

Heung-Yeung Shum, Richard Szeliski,
Simon Baker†, Mei Han‡, and P. Anandan

Microsoft Research, † Columbia University, ‡ Carnegie Mellon University
http://www.research.microsoft.com/research/vision/

Abstract. We present some recent progress in designing and implementing two interactive image-based 3D modeling systems.
The first system constructs 3D models from a collection of panoramic image mosaics. A panoramic mosaic consists of a set of images taken around the same viewpoint, and a camera matrix associated with each input image. The user first interactively specifies features such as points, lines, and planes. Our system recovers the camera pose for each mosaic from known line directions and reference points. It then constructs the 3D model using all available geometrical constraints.
The second system extracts structure from stereo by representing the scene as a collection of approximately planar layers. The user first interactively segments the images into corresponding planar regions. Our system recovers a composite mosaic for each layer, estimates the plane equation for the layer, and optionally recovers the camera locations as well as out-of-plane displacements.
By taking advantage of known scene regularities, our interactive systems avoid difficult feature correspondence problems that occur in traditional automatic modeling systems. They also shift the interactive high-level structural model specification stage to precede (or intermix with) the 3D geometry recovery. They are thus able to extract accurate wire frame and texture-mapped 3D models from multiple image sequences.

1 Introduction

A lot of progress has been made recently in developing automated techniques for 3D scene reconstruction from multiple images, both with calibrated and uncalibrated cameras [1–9]. Unfortunately, the results from many automated modeling systems are disappointing due to the complexity of real scenes and the fragility of fully automated vision techniques. Part of the reason stems from the accurate and robust correspondences required by many computer vision techniques such as stereo and structure from motion. Moreover, such correspondences may not be available in regions of the scene that are untextured.

Automated techniques often require manual clean-up and post-processing to segment the scene into coherent objects and surfaces, or to triangulate sparse point matches [5]. They may also be required to enforce geometric constraints

such as known orientations of surfaces. For instance, building interiors and exteriors provide vertical and horizontal lines and parallel and perpendicular planes. In this paper, we attack the 3D modeling problem from the other side: we specify some geometric knowledge ahead of time (e.g., known orientations of lines, co-planarity of points, initial scene segmentations), and use these constraints to guide our matching and reconstruction algorithms.

The idea of using geometric constraints has previously been exploited in several interactive modeling systems. For example, PhotoModeler [10] is a commecial product which constructs 3D models from several images, using photogrammetry techniques and manually specified points. The TotalCalib system, on the other hand, estimates the fundamental matrix from a few hand-matched points, and then predicts and verifies other potential matching points [11]. The Facade system exploits the known rectahedral structure of building exteriors to directly recover solid 3D models (blocks) from multiple images [12].

This paper presents two interactive (semi-automated) systems for recovering 3D models of large-scale environments from multiple images. Our first system uses one or more panoramic image mosaics, i.e., collections of images taken from the same viewpoint that have been registered together[13]. Panoramas offer several advantages over regular images. First, we can decouple the modeling problem into a zero baseline problem (building panoramas from images taken with rotating camera) and a wide baseline stereo or structure from motion problem (recovering 3D model from one or more panoramas). Second, the intrinsic camera calibrations are recovered as part of the panorama construction [14]. Due to recent advances, it is now possible to construct panoramas even with hand-held cameras [15].

Unlike previous work on 3D reconstruction from multiple panoramas [16, 17], our 3D modeling system exploits important regularities present in the environment, such as walls with known orientations. Fortunately, the man-made world is full of constraints such as parallel lines, lines with known directions, planes with lines and points on them. Using these constraints, we can construct a fairly complex 3D model from a single panorama (or even a wide-angle photograph), and easily handle large co-planar untexture regions such as walls. Using multiple panoramas, more complete and accurate 3D models can be constructed.

Multiple-image stereo matching can be used to recover a more detailed description of surface shape than can be obtained by simply triangulating matched feature points [8]. Unfortunately, stereo fails in regions without texture. A simple depth map also cannot capture the full complexity of a large-scale environment. Methods for overcoming this limitation include volumetric stereo techniques [18, 19] and model-based stereo [12].

In this paper, we propose a different approach—extending the concept of layered motion estimates [20, 21] and "shallow" objects [22] to true multi-image stereo matching. Our second interactive modeling system reconstructs the 3D scene as a collection of approximately planar layers, each of which has an explicit 3D plane equation, a color image with per-pixel opacity, and optionally a per-pixel out-of-plane displacement[23]. This representation allows us to account for

inter-surface occlusions, which traditional stereo systems have trouble modeling correctly.

2 3D modeling from panoramas

2.1 Interactive modeling system

Our modeling system uses one or more panoramas. For each panorama, we draw points, lines, and planes, set appropriate properties for them, and then recover the 3D model. These steps can be repeated in any order to refine or modify the model. The modeling system attempts to satisfy all possible constraints in a consistent and coherent way.

Three coordinate systems are used in our work. The first is the world coordinate system where the 3D model geometry is defined. The second is the "2D" camera coordinate system (panorama coordinates). The third is the screen coordinate system where zoom and rotation (pan and tilt, but no roll) are applied to facilitate user interaction. While each panorama has a single 2D coordinate system, several views of a given panorama can be open simultaneously, each with its own screen coordinate system.

We represent the 3D model by a set of points, lines and planes. Each point is represented by its 3D coordinate $\mathbf{x}$. Each line is represented by its line direction $\mathbf{m}$ and points on the line. Each plane is defined by $(\mathbf{n}, d)$ where $\mathbf{n}$ is the normal, d is the distance to the origin, and $\mathbf{n} \cdot \mathbf{x} + d = 0$ or $(\mathbf{n}, d) \cdot (\mathbf{x}, 1) = 0$. A plane typically includes several vertices and lines.

Each 2D model consists of a set of 2D points and lines extracted from a panorama. A panorama consists of a collection of images and their associated transformations. A 2D point $\tilde{\mathbf{x}}$ (i.e., on a panorama) represents a ray going through the 2D model origin (i.e., camera optical center).[1] Likewise, a 2D line (represented by its line direction $\tilde{\mathbf{m}}$) lies on the "line projection plane" (with normal $\tilde{\mathbf{n}}_p$) which passes through the line and 2D model origin.[2]

2.2 Modeling steps

Many constraints exist in real scenes. For example, we may have known quantities like points, lines, and planes. Or we may have known relationships such as parallel and vertical lines and planes, points on a line or a plane. With multiple panoramas, we have more constraints from corresponding points, lines, and planes.

Some of these constraints are bilinear. For example, a point on a plane is a bilinear constraint in both the point location and the plane normal. However, plane normals and line directions can be recovered without knowing plane distance and

[1] We use the notation $\tilde{\mathbf{x}}$ for a 2D point, $\mathbf{x}$ for a 3D point, and $\hat{\mathbf{x}}$ for a 3D point whose position is known. Likewise for line directions, plane normals, etc..

[2] If a pixel has the screen coordinate $(u, v, 1)$, its 2D point on the panorama is represented by (u, v, f) where f is the focal length.

points. Thus, in our system we decouple the modeling process into several linear steps: (a) recovering camera orientations ($\mathbf{R}$) from known line directions; (b) recovering camera translations ($\mathbf{t}$) from known points; (c) estimating plane normals ($\mathbf{n}$) and line directions ($\mathbf{m}$); (d) estimating plane distances (d), vertex positions ($\mathbf{x}$). These steps are explained in detail in the next sections.

2.3 Recovering camera pose

The camera poses describe the relationship between the 2D models (panorama coordinate systems) and the 3D model (world coordinate system).

To recover the camera rotation, we use lines with known directions. For example, one can easily draw several vertical lines at the intersections of walls and mark them to be parallel to the Z axis of the world coordinate system. Given at least two vertical lines and a horizontal line, or two sets of parallel lines of known directions, the camera matrix can be recovered. This is achieved by computing vanishing points for the parallel lines, and using these to constrain the rotation matrix. If more than 2 vanishing points are available, a least squares solution can be found for $\mathbf{R}$.

To recover the translation, observe that a point on a 2D model (panorama) represents a ray from the camera origin through the pixel on the image,

$$(\mathbf{x} - \mathbf{t}) \times \mathbf{R}^T \tilde{\mathbf{x}} = 0. \tag{1}$$

This is equivalent to

$$(\mathbf{x} - \mathbf{t}) \cdot (\mathbf{R}^T \tilde{\mathbf{p}}_j) = 0, j = 0, 1, 2, \tag{2}$$

where $\tilde{\mathbf{p}}_0 = (-x_2, x_1, 0)$, $\tilde{\mathbf{p}}_1 = (-x_3, 0, x_1)$ and $\tilde{\mathbf{p}}_2 = (0, -x_3, x_2)$ are three directions perpendicular to the ray $\tilde{\mathbf{x}} = (x_1, x_2, x_3)$. Note that only two of the three constraints are linearly independent.[3] Thus, camera translation $\mathbf{t}$ can be recovered as a linear least-squares problem if we have two or more given points. Given a single known point, $\mathbf{t}$ can be recovered only up to a scale. In practice, it is convenient to fix a few points in 3D model, such as the origin $(0, 0, 0)$. These given points are also used to eliminate the ambiguities in recovering camera pose.

For a single panorama, the translation $\mathbf{t}$ is set to zero if no point in 3D model is given. This implies that the camera coordinate coincides with the 3D model coordinate.

2.4 Estimating plane normals

Once we have camera pose, we can recover the scene geometry. Because of the bilinear nature of some constraints (such as points on planes), we recover plane normals ($\mathbf{n}$) before solving for plane distances (d) and points ($\mathbf{x}$). If a normal is given (north, south, up, down, etc.), it can be enforced as a hard constraint.

[3] The third constraint with minimum $\|\tilde{\mathbf{p}}_i\|^2$ is eliminated.

Otherwise, we compute the plane normal $\mathbf{n}$ by finding two line directions on the plane.

If we draw two pairs of parallel lines (a parallelogram) on a plane, we can recover the plane normal. Because $\mathbf{R}$ has been estimated, and we know how to compute a line direction (i.e., the vanishing point $\tilde{\mathbf{m}}$) from two parallel lines, we obtain $\mathbf{m} = \mathbf{R}^T\tilde{\mathbf{m}}$. From two line directions $\mathbf{m}_1$ and $\mathbf{m}_2$ on a plane, the plane normal can be computed as $\mathbf{n} = \mathbf{m}_1 \times \mathbf{m}_2$.

In general, the line direction recovery problem can be formulated as a standard minimum eigenvector problem. Because each "line projection plane" is perpendicular to the line (i.e., $\tilde{\mathbf{n}}_{pi} \cdot \tilde{\mathbf{m}} = 0$), we want to minimize

$$e = \sum_i (\tilde{\mathbf{n}}_{pi} \cdot \tilde{\mathbf{m}})^2 = \tilde{\mathbf{m}}^T (\sum_i \tilde{\mathbf{n}}_{pi}\tilde{\mathbf{n}}_{pi}^T)\tilde{\mathbf{m}}. \tag{3}$$

This is equivalent to finding the vanishing point of the lines [24]. The advantage of the above formulation is that the sign ambiguity of $\tilde{\mathbf{n}}_{pi}$ can be ignored. When only two parallel lines are given, the solution is simply the cross product of two line projection plane normals.

Using the techniques described above, we can therefore recover the surface orientation of an arbitrary plane (e.g., tilted ceiling) provided either we can draw a parallelogram (or a 3-sided rectangle) on the plane.

2.5 Estimating the 3D model

Given camera pose, line directions, and plane normals, recovering plane distances (d), 3D points ($\mathbf{x}$), and camera translation $\mathbf{t}$ (if desired), can be formulated as a linear system consisting of all possible constraints. By separating hard constraints from soft ones, we obtain a least-squares system with equality constraints. Intuitively, the difference between soft and hard constraints is their weights in the least-squares formulation. Soft constraints have unit weights, while hard constraints have very large weights [25].

Some constraints (e.g., a point is known) are inherently hard, therefore equality constraints. Some constraints (e.g., a feature location on a 2D model or panorama) are most appropriate as soft constraints because they are based on noisy image measurements. Take a point on a plane for an example. If the plane normal $\hat{\mathbf{n}}_k$ is given, we consider the constraint $(\mathbf{x}_i \cdot \hat{\mathbf{n}}_k + d_k = 0)$ as hard. We use the notations $\hat{\mathbf{m}}$ and $\hat{\mathbf{n}}$ to represent the given line direction $\mathbf{m}$ and plane normal $\mathbf{n}$, respectively. This implies that the point has to be on the plane, only its location can be adjusted. On the other hand, if the plane normal $\mathbf{n}_k$ is estimated, we consider the constraint $(\mathbf{x}_i \cdot \mathbf{n}_k + d_k = 0)$ as soft. This could lead to an estimated point that is not on the plane at all. So why not make the constraint $(\mathbf{x}_i \cdot \mathbf{n}_k + d_k = 0)$ hard as well?

The reason is that we may end up with a very bad model if some of the estimated normals have large errors. Too many hard constraints could conflict with one another or make other soft constraints insignificant. To satisfy all possible constraints, we formulate our modeling process as an equality-constrained least-squares problem. In other words, we would like to solve the linear system (soft

constraints) $\mathbf{Ax} = \mathbf{b}$ subject to (hard constraints) $\mathbf{Cx} = \mathbf{q}$ where $\mathbf{A}$ is $m \times n$, $\mathbf{C}$ is $p \times n$. A solution to the above problem is to use the QR factorization [25].

Before we can apply the equality-constrained linear system solver, we must check whether the linear system formed by all constraints is solvable. In general, the system may consist of several subsystems (connected components) which can be solved independently. For example, when modeling a room with a computer monitor floating in the space not connected with any wall, ceiling or floor, we may have a system with two connected components. To find all connected components, we use depth first search to step through the linear system. For each connected components we check that: (a) the number of equations (including both hard and soft constraints) is no fewer than the number of unknowns; (b) the right hand side is a non-zero vector, i.e., has some minimal ground truth data; (c) the hard constraints are consistent. If any of the above is not satisfied, the system is declared unsolvable, and a warning message is then generated to indicate which set of unknowns cannot be recovered.

3 3D modeling using layered stereo

3.1 Overview of layered stereo approach

Our second 3D modeling system interactively extracts structure as a collection of 3D (quasi-) planar layers from multiple images. The basic concepts of the layered stereo approach are illustrated in Figure 1. Assume that we are given as input K images $I_1(\mathbf{u}_1), I_2(\mathbf{u}_2), \ldots, I_K(\mathbf{u}_K)$[4] captured by K cameras with camera matrices $\mathbf{P}_1, \mathbf{P}_2, \ldots, \mathbf{P}_K$. In what follows, we will drop the image coordinates $\mathbf{u}_k$ unless they are needed to explain a warping operation explicitly. Our hypothesis is that we can reconstruct the world as a collection of L approximately planar layers. Following [26], we denote a layer "sprite" image by $L_l(\mathbf{u}_l) = (\alpha_l \cdot r_l, \alpha_l \cdot g_l, \alpha_l \cdot b_l, \alpha_l)$, where $r_l = r_l(\mathbf{u}_l)$ is the red band, $g_l = g_l(\mathbf{u}_l)$ is the green band, $b_l = b_l(\mathbf{u}_l)$ is the blue band, and $\alpha_l = \alpha_l(\mathbf{u}_l)$ is the opacity of pixel $\mathbf{u}_l$.[5] We also associate with each layer a homogeneous vector $\mathbf{n}_l$ which defines the plane equation of the layer via $\mathbf{n}_l^T\mathbf{x} = 0$, and optionally a per-pixel residual depth offset $Z_l(\mathbf{u}_l)$.

A number of automatic techniques have been developed to initialize the layers, e.g, merging [20, 28, 29], splitting [30, 28], color segmentation [31] and plane fitting to a recovered depth map. In our system, we interactively initialize the layers because we wish to focus initially on techniques for creating composite (mosaic) sprites from multiple images, estimating the sprite plane equations, and refining layer assignments. We plan to incorporate automated initialization techniques later.

[4] We use homogeneous coordinates in this section for both 3D world coordinates $\mathbf{x} = (x, y, z, 1)^T$ and for 2D image coordinates $\mathbf{u} = (u, v, 1)^T$.

[5] The terminology comes from computer graphics, where sprites are used to quickly (re-)render scenes composed of many objects [27].

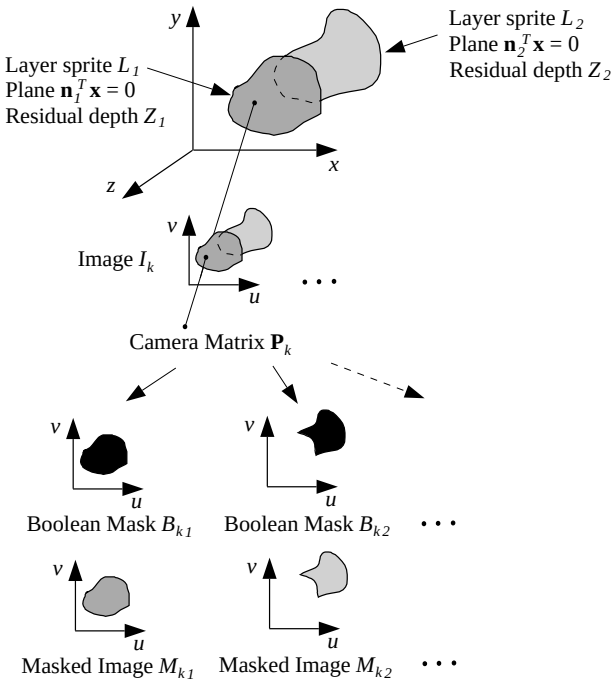

Fig. 1. Suppose K images I_k are captured by K cameras $\mathbf{P}_k$. We assume the scene can be represented by L sprite images L_l on planes $\mathbf{n}_l^T\mathbf{x} = 0$ with depth offsets Z_l. The boolean masks B_{kl} denote the pixels in image I_k from layer L_l and the masked images $M_{kl} = B_{kl} \cdot I_k$.

The input consists of a collection of images I_k taken with known camera matrices $\mathbf{P}_k$. The camera matrices can be estimated when they are not known *a priori*, either using traditional structure from motion [3, 7, 8], or directly from the homographies relating sprites in different images [32, 33]. Our goal is to estimate the layer sprites L_l, the plane vectors $\mathbf{n}_l$, and the residual depths Z_l.

Our approach can be subdivided into a number of steps where we estimate each of L_l, $\mathbf{n}_l$, and Z_l in turn. To compute these quantities, we use auxiliary boolean mask images B_{kl}. The boolean masks B_{kl} denote the pixels in image I_k which are images of points in layer L_l. Since we are assuming boolean opacities, $B_{kl} = 1$ if and only if L_l is the front-most layer which is opaque at that pixel in image I_k. Hence, in addition to L_l, $\mathbf{n}_l$, and Z_l, we also need to estimate the boolean masks B_{kl}. Once we have estimated these masks, we can compute masked input images $M_{kl} = B_{kl} \cdot I_k$ (see Figure 1).

Given any three of L_l, $\mathbf{n}_l$, Z_l, and B_{kl}, there are techniques for estimating the remaining one. Our algorithm therefore consists of first initializing these quantities. Then, we iteratively estimate each of theses quantities in turn fixing the other three.

3.2 Estimation of Plane Equations

In order to compute the plane equation vector $\mathbf{n}_l$, we need to be able to map points in masked image M_{kl} onto the plane $\mathbf{n}_l^T\mathbf{x} = 0$. If $\mathbf{x}$ is a 3D world coordinate of a point and $\mathbf{u}_k$ is the image of $\mathbf{x}$ in camera $\mathbf{P}_k$, we have:

$$\mathbf{u}_k \;=\; \mathbf{P}_k\mathbf{x}$$

where equality is in the 2D projective space $\mathcal{P}^2$. Since $\mathbf{P}_k$ is of rank 3, we can write:

$$\mathbf{x} \;=\; \mathbf{P}_k^*\mathbf{u}_k + s\mathbf{p}_k \tag{4}$$

where $\mathbf{P}_k^* = \mathbf{P}_k^T(\mathbf{P}_k\mathbf{P}_k^T)^{-1}$ is the pseudoinverse of $\mathbf{P}_k$, s is an unknown scalar, and $\mathbf{p}_k$ is a vector in the null space of $\mathbf{P}_k$, i.e. $\mathbf{P}_k\mathbf{p}_k = 0$. If $\mathbf{x}$ lies on the plane $\mathbf{n}_l^T\mathbf{x} = 0$ we can solve for s, substitute into Equation (4), and obtain:

$$\mathbf{x} \;=\; \left((\mathbf{n}_l^T\mathbf{p}_k)\mathbf{I} - \mathbf{p}_k\mathbf{n}_l^T\right)\mathbf{P}_k^*\mathbf{u}_k. \tag{5}$$

Equation (5) allows us to map a point $\mathbf{u}_k$ in image M_{kl} onto the point on plane $\mathbf{n}_l^T\mathbf{x} = 0$, of which it is an image. Afterwards we can map this point onto its image in another camera $\mathbf{P}_{k'}$:

$$\mathbf{u}_{k'} \;= \mathbf{P}_{k'}\left((\mathbf{n}_l^T\mathbf{p}_k)\mathbf{I} - \mathbf{p}_k\mathbf{n}_l^T\right)\mathbf{P}_k^*\mathbf{u}_k \;\equiv\; \mathbf{H}_{kk'}^l\mathbf{u}_k \tag{6}$$

where $\mathbf{H}_{kk'}^l$ is a homography (collineation of $\mathcal{P}^2$). Equation (6) describes the image coordinate warp between the two images M_{kl} and $M_{k'l}$ which would hold if all the masked image pixels were images of world points on the plane $\mathbf{n}_l^T\mathbf{x} = 0$. Using this relation, we can warp all of the masked images onto the coordinate frame of one distinguished image, w.l.o.g. image M_{1l}, as follows:

$$\left(\mathbf{H}_{1k}^l \circ M_{kl}\right)(\mathbf{u}_1) \;\equiv\; M_{kl}\left(\mathbf{H}_{1k}^l\mathbf{u}_1\right).$$

Here, $\mathbf{H}_{1k}^l \circ M_{kl}$ is the masked image M_{kl} warped into the coordinate frame of M_{1l}.

We can therefore solve for $\mathbf{n}_l$ by finding the value for which the homographies $\mathbf{H}_{1k}^l$ defined in Equation (6) best register the images onto each other. Typically, this value is found using some form of gradient decent, such as the Gauss-Newton method, and the optimization is performed in a hierarchical (i.e. pyramid based) fashion to avoid local extrema [34]. To apply this approach, we compute the Jacobian of the image warp $\mathbf{H}_{1k}^l$ with respect to the parameters of $\mathbf{n}_l$. Alternatively, we can first compute a set of unconstrained homographies using a standard mosaic construction algorithm, and then invoke a structure from motion algorithm to recover the plane equation (and also the camera matrices, if desired) [32, 33].

3.3 Estimation of Layer Sprites

Before we can compute the layer sprite images L_l, we need to choose 2D coordinate systems for the planes. Such coordinate systems can be specified by a

collection of arbitrary (rank 3) camera matrices $\mathbf{Q}_l$.[6] Then, following the same argument used in Equations (5) and (6), we can show that the image coordinates $\mathbf{u}_k$ of the point in image M_{kl} which is projected onto the point $\mathbf{u}_l$ on the plane $\mathbf{n}_l^T\mathbf{x} = 0$ is given by:

$$\mathbf{u}_k \;=\; \mathbf{P}_k \left((\mathbf{n}_l^T \mathbf{q}_l)\mathbf{I} - \mathbf{q}_l \mathbf{n}_l^T \right) \mathbf{Q}_l^* \mathbf{u}_l \;\equiv\; \mathbf{H}_k^l \mathbf{u}_k \tag{7}$$

where $\mathbf{Q}_l^*$ is the pseudo-inverse of $\mathbf{Q}_l$ and $\mathbf{q}_l$ is a vector in the null space of $\mathbf{Q}_l$. The homography $\mathbf{H}_k^l$ can be used to warp the image M_{kl} forward onto the plane, the result of which is denoted $\mathbf{H}_k^l \circ M_{kl}$. After we have warped the masked image onto the plane, we can estimate the layer sprite (with boolean opacities) by "blending" the warped images:

$$L_l \;=\; \bigoplus_{k=1}^{K} \mathbf{H}_k^l \circ M_{kl} \tag{8}$$

where $\bigoplus$ is a blending operator.

There are a number of ways the blending can be performed. One simple method is to take the mean of the color or intensity values. A refinement is to use a "feathering" algorithm, where the averaging is weighted by the distance of each pixel from the nearest invisible pixel in M_{kl} [35]. Alternatively, robust techniques can be used to estimate L_l from the warped images.

3.4 Estimation of Residual Depth

In general, the scene will not be exactly piecewise planar. To model any non-planarity, we assume that the point $\mathbf{u}_l$ on the plane $\mathbf{n}_l^T\mathbf{x} = 0$ is displaced slightly in the direction of the ray through $\mathbf{u}_l$ defined by the camera matrix $\mathbf{Q}_l$, and that the distance it is displaced is $Z_l(\mathbf{u}_l)$, measured in the direction normal to the plane. In this case, the homographic warps used in the previous section are not applicable. However, using a similar argument to that in Sections 3.2 and 3.3, it is easy to show (see also [36, 12]) that:

$$\mathbf{u}_k \;=\; \mathbf{H}_k^l \mathbf{u}_l + Z_l(\mathbf{u}_l)\mathbf{t}_{kl} \tag{9}$$

where $\mathbf{H}_k^l = \mathbf{P}_k \left((\mathbf{n}_l^T \mathbf{q}_l)\mathbf{I} - \mathbf{q}_l \mathbf{n}_l^T \right) \mathbf{Q}_l^*$ is the planar homography of Section 3.3, $\mathbf{t}_{kl} = \mathbf{P}_k \mathbf{q}_l$ is the epipole, and it is assumed that the plane equation vector $\mathbf{n}_l = (n_x, n_y, n_z, n_d)^T$ has been normalized so that $n_x^2 + n_y^2 + n_z^2 = 1$. Equation (9) can be used to map plane coordinates $\mathbf{u}_l$ backwards to image coordinates $\mathbf{u}_k$, or to map the image M_{kl} forwards onto the plane. We denote the result of this warp by $(\mathbf{H}_k^l, \mathbf{t}_{kl}, Z_l) \circ M_{kl}$, or $\mathbf{W}_k^l \circ M_{kl}$ for more concise notation.

To compute the residual depth map Z_l, we could optimize the same (or a similar) consistency metric as that used in Section 2.2 to estimate the plane equation. Doing so is essentially solving a simpler (or what [12] would call "model-based") stereo problem. In fact, almost any stereo algorithm could be used to compute Z_l. The algorithm should favor small disparities.

[6] A reasonable choice for $\mathbf{Q}_l$ is one of the camera matrices $\mathbf{P}_k$.

3.5 Pixel Assignment to Layers

In the previous three sections, we have assumed a known assignment of pixels to layer, i.e., known boolean masks B_{kl} which allow us to compute the masked image M_{kl} using $M_{kl} = B_{kl} \cdot I_k$. We now describe how to estimate the pixel assignments from $\mathbf{n}_l$, L_l, and Z_l.

We could try to update the pixel assignments by comparing the warped images $\mathbf{W}_k^l \circ M_{kl}$ to the layer sprite images L_l. However, if we compared these images, we would not be able to deduce anything about the pixel assignments outside of the current estimates of the masked regions. To allow the boolean mask B_{kl} to "grow", we therefore compare $\mathbf{W}_k^l \circ I_k$ with:

$$\tilde{L}_l \;=\; \bigoplus_{k=1}^{K} \mathbf{W}_k^l \circ \tilde{M}_{kl},$$

where $\tilde{M}_{kl} = \tilde{B}_{kl} \cdot I_k$ and $\tilde{B}_{kl}$ is a dilated version of B_{kl} (if necessary, Z_l is also enlarged so that it declines to zero outside the masked region).

Given the enlarged layer sprites $\tilde{L}_l$, our approach to pixel assignment is as follows. We first compute a measure $P_{kl}(\mathbf{u}_l)$ of the likelihood that the pixel $\mathbf{W}_k^l \circ I_k(\mathbf{u}_l)$ is the warped image of the pixel $\mathbf{u}_l$ in the enlarged layer sprite $\tilde{L}_l$. Next, P_{kl} is warped back into the coordinate system of the input image I_k to yield:

$$\hat{P}_{kl} \;=\; (\mathbf{W}_k^l)^{-1} \circ P_{kl}.$$

This warping tends to blur P_{kl}, but this is acceptable since we will want to smooth the pixel assignment. The pixel assignment can then be computed by choosing the best possible layer for each pixel:

$$B_{lk}(\mathbf{u}_k) \;=\; \begin{cases} 1 \text{ if } \hat{P}_{kl}(\mathbf{u}_k) \;=\; \min_{l'} \hat{P}_{kl'}(\mathbf{u}_k) \\ 0 \text{ otherwise} \end{cases}.$$

The simplest ways of defining P_{kl} is the *residual intensity difference* [28]; another possibility is the residual normal flow magnitude [30]. A third possibility would be to compute the optical flow between $\mathbf{W}_k^l \circ I_k$ and $\tilde{L}_l$ and then use the magnitude of the flow for P_{kl}.

3.6 Layer refinement by re-synthesis

The layered stereo algorithm described above is limited to recovering binary masks B_{kl} for the assignment of input pixels to layers. If we wanted to, we could use an EM (expectation maximization) algorithm to obtain graded (continuous) assignments [37, 38]. However, EM models mixtures of probability distributions, rather than the kind of partial occlusion mixing that occurs at sprite boundaries [26]. Stereo techniques inspired by matte extraction [39] are needed to refine the color/opacity estimates for each layer [19]. Such an algorithm would work by re-synthesizing each input image from the current sprite estimates, and then adjusting pixel colors and opacities so at to minimize the difference between the original and re-synthesized images. We are planning to implement such an algorithm in future work.

4 Experiments

We have implemented our panorama 3D modeling system on a PC and tested it with single and multiple panoramas. The system consists of two parts: the interface (viewing the panorama with pan, tilt, and zoom control) and the modeler (recovering the camera pose and the 3D model). Figure 2 shows a spherical panoramic image on the left and a simple reconstructed 3D model on the right. The coordinate system on the left corner (red) is the world coordinate, and the coordinate system in the middle (green) is the camera coordinate. The panorama is composed of 60 images using the method of creating full-view panoramas [35]. The extracted texture maps (without top and bottom faces) are shown in Figure 3. Notice how the texture maps in Figure 3 have different sampling rates from the original images. The sampling is the best (e.g., Figure 3(b)) when the surface normal is parallel with the viewing direction from the camera center, and the worst (e.g., Figure 3(d)) when perpendicular. This explains why the sampling on the left is better than that on the right in Figure 3(a). Figure 4 shows two views of our interactive modeling system. Green lines and points are the 2D items that are manually drawn and assigned with properties, and blue lines and points are projections of the recovered 3D model. It took about 15 minutes for the authors to build the simple model in Figure 2. In 30 minutes, we can construct the more complicated model shown in Figure 5.

Figures 6 and 7 show an example of building 3D models from multiple panoramas. Figure 6 shows two spherical panoramas built from image sequences taken with a hand-held digital video camera. Figure 7 shows two views of reconstructed 3D wireframe model from the two panoramas in Figure 6. Notice that the occluded middle area in the first panorama (behind the tree) is recovered because it is visible in the second panorama.

We have applied our layered stereo modeling system to a number of multi-frame stereo data sets. A standard point tracking and structure from motion algorithm is used to recover a camera matrix for each image. To initialize our algorithm, we interactively specify how many layers and then perform a rough assignment of pixels to layers. Next, an automatic hierarchical parametric motion estimation algorithm similar to [34] is used to find the homographies between the layers, as defined in Equation (7). For the experiments presented in this paper, we set $\mathbf{Q}_l = \mathbf{P}_1$, i.e. we reconstruct the sprites in the coordinate system of the first camera. Using these homographies, we find the best plane estimate for each layer using a Euclidean structure from motion algorithm [40].

The results of applying these steps to the MPEG *flower garden* sequence are shown in Figure 8. Figures 8(a) and (b) show the first and last image in the subsequence we used (the first seven even images). Figure 8(c) shows the initial pixel labeling into seven layers. Figures 8(d) and (e) show the sprite images corresponding to each of the seven layers, re-arranged for more compact display. Note that because of the compositing and blending that takes place during sprite construction, each sprite is larger than its footprint in any one of the input images. This sprite representation makes it very easy to re-synthesize novel images without leaving gaps in the new image, unlike approaches based on a single

Fig. 2. 3D model from a single panorama.

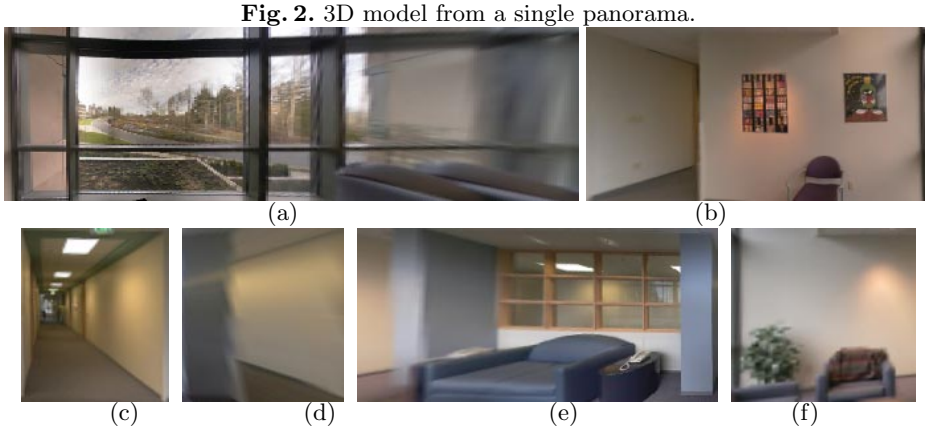

Fig. 3. Texture maps for the 3D model.

Fig. 4. Two views of the interactive system.

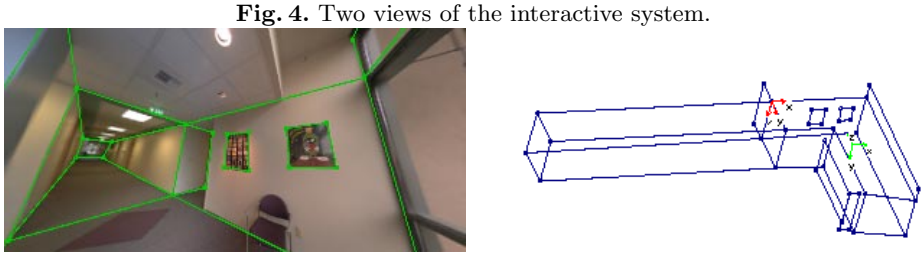

Fig. 5. A more complex 3D model from a single panorama.

Fig. 6. Two input panoramas of an indoor scene.

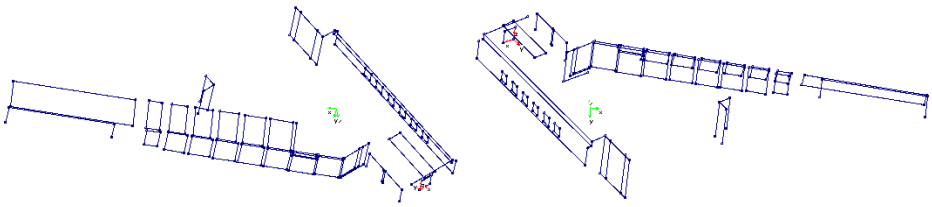

Fig. 7. Two views of a 3D model from multiple panoramas.

painted depth map [16]. Figure 8(f) shows the depth map computed by painting every pixel in every sprite with its corresponding color coded Z value, and then re-compositing the image. Notice how the depth discontinuities are much crisper and cleaner than those available with traditional stereo correspondence algorithms.

Our second set of experiments uses five images taken from a 40-image stereo dataset taken at a computer graphics symposium. Figure 9(a) shows the middle input image, Figure 9(b) shows the initial pixel assignment to layers, and Figure 9(c) shows the recovered depth map. Figures 9(d) and (e) show the recovered sprites, and Figure 9(f) shows the middle image re-synthesized from these sprites. The gaps visible in Figures 9(c) and 9(f) lie *outside* the area corresponding to the middle image, where the appropriate parts of the background sprites could not be seen.

5 Discussion and conclusions

In this paper, we have presented two systems for interactively constructing complex (large-scale) 3D models from multiple images. Our modeling systems are able to construct accurate geometrical and photo-realistic 3D models because our approaches have much less ambiguity than traditional structure from motion or stereo approaches. Our results show that it is desirable and practical for the modeling systems to take advantage of as many regularities and priori knowledge about man-made environments (such as vertices, lines, and planes) as possible [41].

Our panorama 3D modeling system decomposes the modeling process into a zero baseline problem (panorama construction) and a wide baseline problem (stereo or structure from motion). Using the knowledge of the scene, e.g., known line directions, parallel or perpendicular planes, our system first recovers the camera pose for each panorama, and then constructs the 3D model using all possible constraints. In particular, we carefully partition the recovery problem into a series of linear estimation stages, and divide the constraints into "hard" and "soft" constraints so that each estimation stage becomes a linearly-constrained least-squares problem.

Our layered stereo modeling system makes use of a different kind of scene regularity, where input images are segmented into planar layers (with possible

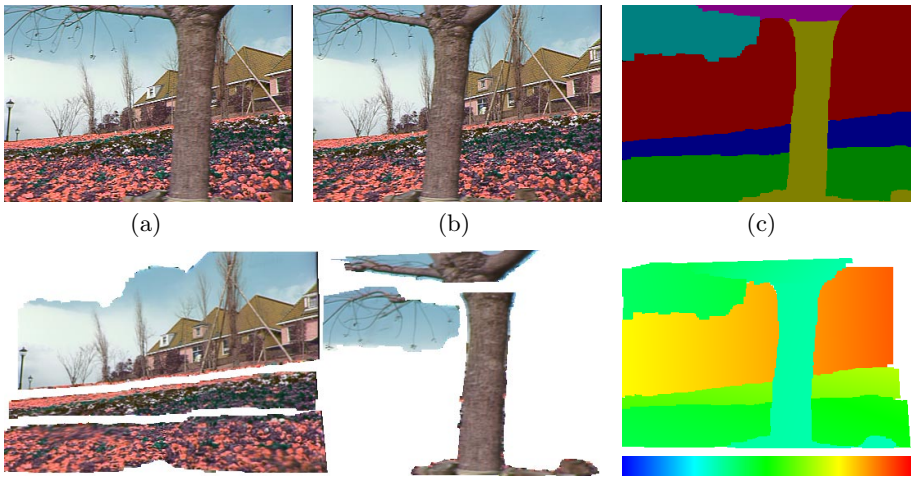

Fig. 8. Results on the *flower garden* sequence: (a) first and (b) last input images; (c) initial segmentation into six layers; (d) and (e) the six layer sprites; (f) depth map for planar sprites (bottom strip illustrates the coding of depths as colors)

Fig. 9. Results on the *symposium* sequence: (a) third of five images; (b) initial segmentation into six layers; (c) recovered depth map; (d) and (e) the five layer sprites; (f) re-synthesized third image (note extended field of view).

small depth offsets). By drastically reducing the number of unknowns (only 3 parameters for each sprite plane), the recovery of 3D structure is much more robust than conventional stereo algorithms.

We are working on several extensions to improve the usability and generality of our system. Naturally, we want to automate even more parts of the interactive systems. For the panorama modeling system, we have implemented an automatic line snapping technique which snaps lines to their closest edges present in the panorama. We also plan to incorporate automatic line detection, corner detection as well as inter-image correspondence and other feature detections to further automate the system. If we use more features with automatic feature extraction and correspondence techniques, robust modeling techniques should also be developed [7]. For the layered stereo modeling system, we plan to automate the interactive masking process by only specifying layers in few (e.g., the first and the last) images and by incorporating motion segmentation and color segmentation techniques.

We are also planning to combine our layered stereo modeling system with the panorama modeling system. The idea is to build a rough model using panorama modeling system and refine it using layered stereo wherever it is appropriate (similar in spirit to the model-based stereo of [12]). We are also investigating representations beyond texture-mapped 3D models, i.e, image-based rendering approaches [42] such as view-dependent texture maps [12] and layered depth images [43]. Integrating all of these into one interactive modeling system will enable users to easily construct complex photorealistic 3D models from images.

References

1. O. D. Faugeras. What can be seen in three dimensions with an uncalibrated stereo rig? In *Second European Conference on Computer Vision (ECCV'92)*, pages 563–578, Santa Margherita Liguere, Italy, May 1992. Springer-Verlag.
2. R. Mohr, L. Veillon, and L. Quan. Relative 3D reconstruction using multiple uncalibrated images. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'93)*, pages 543–548, New York, New York, June 1993.
3. O. Faugeras. *Three-dimensional computer vision: A geometric viewpoint*. MIT Press, Cambridge, Massachusetts, 1993.
4. A. Shashua. Projective structure from uncalibrated images: Structure from motion and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(8):778–790, August 1994.
5. A. Azarbayejani and A. P. Pentland. Recursive estimation of motion, structure, and focal length. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(6):562–575, June 1995.
6. O. D. Faugeras, Laveau S., Robert L., Csurka G., and Zeller C. 3-D reconstruction of urban scenes from sequences of images. *Computer Vision and Image Understanding*, 69(3):292–309, March 1998.
7. P. Beardsley, P. Torr, and A. Zisserman. 3D model acquisition from extended image sequences. In *Fourth European Conference on Computer Vision (ECCV'96)*, volume 2, pages 683–695, Cambridge, England, April 1996. Springer-Verlag.

8. M. Pollefeys, R. Koch, and L. Van Gool. Self-calibration and metric reconstruction in spite of varying and unknown internal camera parameters. In *Sixth International Conference on Computer Vision (ICCV'98)*, pages 90–95, Bombay, January 1998.
9. P. Torr, A. W. Fitzgibbon, and A. Zisserman. Maintaining multiple motion model hypotheses over many views to recover matching structure. In *Sixth International Conference on Computer Vision (ICCV'98)*, pages 485–491, Bombay, January 1998.
10. Photomodeler. //www.photomodeler.com.
11. S. Bougnoux and L. Robert. Totalcalib: a fast and reliable system for off-line calibration of image sequences. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'97)*, June 1997. The Demo Session.
12. P. E. Debevec, C. J. Taylor, and J. Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. *Computer Graphics (SIGGRAPH'96)*, pages 11–20, August 1996.
13. H.-Y. Shum, M. Han, and R. Szeliski. Interactive construction of 3d models from panoramic mosaics. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'98)*, pages 427–433, Santa Barbara, June 1998.
14. R. I. Hartley. Self-calibration from multiple views of a rotating camera. In *Third European Conference on Computer Vision (ECCV'94)*, volume 1, pages 471–478, Stockholm, Sweden, May 1994. Springer-Verlag.
15. R. Szeliski and H.-Y. Shum. Creating full view panoramic image mosaics and texture-mapped models. *Computer Graphics (SIGGRAPH'97)*, pages 251–258, August 1997.
16. L. McMillan and G. Bishop. Plenoptic modeling: An image-based rendering system. *Computer Graphics (SIGGRAPH'95)*, pages 39–46, August 1995.
17. S. B. Kang and R. Szeliski. 3-D scene data recovery using omnidirectional multibaseline stereo. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'96)*, pages 364–370, San Francisco, California, June 1996.
18. S. M. Seitz and C. M. Dyer. Photorealistic scene reconstrcution by space coloring. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'97)*, pages 1067–1073, San Juan, Puerto Rico, June 1997.
19. R. Szeliski and P. Golland. Stereo matching with transparency and matting. In *Sixth International Conference on Computer Vision (ICCV'98)*, pages 517–524, Bombay, January 1998.
20. J. Y. A. Wang and E. H. Adelson. Layered representation for motion analysis. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'93)*, pages 361–366, New York, New York, June 1993.
21. Y. Weiss. Smoothness in layers: Motion segmentation using nonparametric mixture estimation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'97)*, pages 520–526, San Juan, Puerto Rico, June 1997.
22. H. S. Sawhney and A. R. Hanson. Identification and 3D description of 'shallow' environmental structure over a sequence of images. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'91)*, pages 179–185, Maui, Hawaii, June 1991. IEEE Computer Society Press.
23. S. Baker, R. Szeliski, and P. Anandan. A layered approach to stereo reconstruction. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'98)*, pages 434–441, Santa Barbara, June 1998.
24. R. T. Collins and R. S. Weiss. Vanishing point calculation as a statistical inference on the unit sphere. In *Third International Conference on Computer Vision*

(ICCV'90), pages 400–403, Osaka, Japan, December 1990. IEEE Computer Society Press.
25. G. Golub and C. F. Van Loan. *Matrix Computation, third edition*. The John Hopkins University Press, Baltimore and London, 1996.
26. J. F. Blinn. Jim Blinn's corner: Compositing, part 1: Theory. *IEEE Computer Graphics and Applications*, 14(5):83–87, September 1994.
27. J. Torborg and J. T. Kajiya. Talisman: Commodity realtime 3D graphics for the PC. In *Computer Graphics Proceedings, Annual Conference Series*, pages 353–363, Proc. SIGGRAPH'96 (New Orleans), August 1996. ACM SIGGRAPH.
28. H.S. Sawhney and S. Ayer. Compact representations fo videos through dominant and multiple motion estimation. *PAMI*, 18(8):814–830, 1996.
29. M. J. Black and A. D. Jepson. Estimating optical flow in segmented images using variable-order parametric models with local deformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(10):972–986, October 1996.
30. M. Irani, P. Anandan, and S. Hsu. Mosiac based representations of video sequences and their applications. In *5th ICCV*, pages 605–611, 1995.
31. S. Ayer, P. Schroeter, and J. Bigün. Segmentation of moving objects by robust parameter estimation over multiple frames. In *3rd ECCV*, pages 316–327, 1994.
32. Q.-T. Luong and O. Faugeras. Determining the fundamental matrix with planes: Instability and new algorithms. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'93)*, pages 489–494, New York, New York, June 1993.
33. R. Szeliski and P. Torr. Geometrically constrained structure from motion: Points on planes. In *European Workshop on 3D Structure from Multiple Images of Large-scale Environments (SMILE)*, Freiburg, Germany, June 1998.
34. J.R. Bergen, P. Anandan, K.J. Hanna, and R. Hingorani. Hierarchical model-based motion estimation. In *2nd ECCV*, pages 237–252, 1992.
35. H.-Y. Shum and R. Szeliski. Panoramic image mosaicing. Technical Report MSR-TR-97-23, Microsoft Research, September 1997.
36. R. Kumar, P. Anandan, and K. Hanna. Direct recovery of shape from multiple views: A parallax based approach. In *Twelfth International Conference on Pattern Recognition (ICPR'94)*, volume A, pages 685–688, Jerusalem, Israel, October 1994. IEEE Computer Society Press.
37. T. Darrell and A. Pentland. Robust estimation of a multi-layered motion representation. In *IEEE Workshop on Visual Motion*, pages 173–178, Princeton, New Jersey, October 1991. IEEE Computer Society Press.
38. A. Jepson and M. J. Black. Mixture models for optical flow computation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'93)*, pages 760–761, New York, New York, June 1993.
39. A. R. Smith and J. F. Blinn. Blue screen matting. In *Computer Graphics Proceedings, Annual Conference Series*, pages 259–268, Proc. SIGGRAPH'96 (New Orleans), August 1996. ACM SIGGRAPH.
40. T. Viéville, C. Zeller, and L. Robert. Using collineations to compute motion and structure in an uncalibrated image sequence. *International Journal of Computer Vision*, 20(3):213–242, 1996.
41. E. L. Walker and M. Herman. Geometric reasoning for constructing 3D scene descriptions from images. *Artificial Intelligence*, 37:275–290, 1988.
42. Workshop on image-based modeling and rendering. //graphics.stanford.edu/im98/, March 1998.
43. J. Shade, S. Gortler, L.-W. He, and R. Szeliski. Layered depth images. In *Computer Graphics (SIGGRAPH'98) Proceedings*, Orlando, July 1998. ACM SIGGRAPH.