

# Interactive Construction of 3D Models from Panoramic Mosaics

Heung-Yeung Shum

Microsoft Research

Mei Han

Carnegie Mellon University

Richard Szeliski

Microsoft Research

## **Abstract**

This paper presents an interactive modeling system that 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 transformation matrix associated with each input image. Our system first recovers the camera pose for each mosaic from known line directions and points, and then constructs the 3D model using all available geometrical constraints. In particular, we carefully partition constraints into “soft” and “hard” linear constraints so that the modeling process can be formulated as a linearly-constrained least-squares problem, which can be solved efficiently using QR factorization. Wire frame and texture-mapped 3D models from single and multiple panoramas are presented.

# 1 Introduction

A great deal of effort has been expended on 3D scene reconstruction from image sequences (with calibrated or uncalibrated cameras) using computer vision techniques. Unfortunately, the results from most automatic modeling systems are disappointing and unreliable due to the complexity of the real scene and the fragility of the vision techniques. Part of the reason is the demand for accurate correspondences (e.g., point correspondence) required by many computer vision techniques such as stereo and structure from motion. Moreover, such correspondences may not be available if the scene consists of large untextured regions.

Fortunately, for many real scenes, it is relatively straightforward to interactively specify corresponding points, or lines, or planes. For example, building interiors and exteriors provide vertical and horizontal lines and parallel and perpendicular planes. These constraints have been exploited in several interactive modeling systems. For example, PhotoModeler<sup>1</sup>, a commercial product which constructs 3D models from several images, using photogrammetry techniques and manually specified points. Explicit camera calibration is therefore necessary. The TotalCalib system<sup>2</sup>, on the other hand, estimates the fundamental matrix from selected few matched points [BR97]. Then it predicts the possible matching points from one image to others. In Becker's modeling system<sup>3</sup>, the problem of lens distortion (encountered in images taken with wide field of view lens) is also considered [BB95]. By employing the known structure of building exteriors, the Facade system<sup>4</sup> directly recovers a given solid 3D model (blocks) from multiple images [TDM96].

Our system differs from previous interactive modeling systems in that we use multiple panoramic image mosaics (therefore large field of view), instead of multiple images (generally small field of view). A panoramic mosaic is a collection of images taken around the same viewpoint. Why use panoramas? 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 camera calibration problem is implicitly recovered as part

---

<sup>1</sup>[www.photomodeler.com](http://www.photomodeler.com)

<sup>2</sup>[www.inria.fr/robotvis/personnel/sbournou](http://www.inria.fr/robotvis/personnel/sbournou)

<sup>3</sup>[sbeck.www.media.mit.edu/people/sbeck](http://sbeck.www.media.mit.edu/people/sbeck)

<sup>4</sup>[www.cs.berkeley.edu/debevec/Research](http://www.cs.berkeley.edu/debevec/Research)

of panorama construction[Ste95, KW97, SK97]. Lastly, it is easy to construct panoramas even with hand-held cameras[SS97b]. Previous work on 3D reconstruction from multiple panoramas[MB95, KS96] suggests that automatic techniques fail to capture important regularities present in the environment, such as walls with known orientations.

With enough constraints (parallel lines, lines with known directions, planes with lines and points on them, etc.), we can construct a fairly complex 3D model even from a single panorama. Of course the recovered model is only up to a scale, unless we have some knowledge of the scene (e.g., length or width of a room). Using multiple panoramas, more complete and accurate 3D models can be constructed.

We introduce our interactive modeling system in Section 2, explain how to estimate camera orientations using known line directions in Section 3, recover line directions and plane normals in Section 4, and recover camera translations in Section 5. Section 6 presents our solution techniques of combining all possible constraints to form a linearly-constrained least-squares problem. Techniques for building models from multiple panoramas are discussed in Section 7. Examples of 3D models from single and multiple panoramas are shown in Section 8. We close the paper with a discussion of how to extend the system.

## 2 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 of interactively drawing geometric items, setting up properties and modeling 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.

### 2.1 Representation

Three coordinate systems are used in our work (Figure 1). The first is the world coordinate system where the 3D model geometry (planes, lines, vertices) is defined. The second is the “2D”<sup>5</sup> camera coordinate system (panorama coordinates). The third is the screen coordinate

---

<sup>5</sup>Each point (or pixel) on the panorama has only two degrees of freedom because its location is not known.

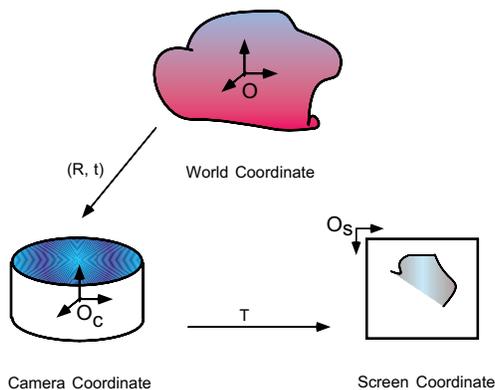


Figure 1: Coordinate systems

system where zoom and rotation (pan and tilt, but no roll) are applied to facilitate the 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. Our system can handle calibrated (non-panoramic) images as well - these are simply treated as simple, narrow field of view panoramas.

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  $(\mathbf{n}, d)(\mathbf{x}, 1) = 0$ . A plane consists of a set of vertices and lines on it.

Each “2D” model associates with the 3D model and is represented by “2D” points, “2D” lines on the panorama. A panorama consists of a collection of images and their transformations. A “2D” point  $\tilde{\mathbf{x}}$ <sup>6</sup> (i.e., on a panorama) represents a ray going through the 2D model origin (i.e., camera optical center). 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 (Figure 2).<sup>7</sup> Therefore, a line direction in a “2D” model can not be uniquely determined by two points in “2D” model.

Note that line directions and plane normals have sign ambiguity. These ambiguity prob-

<sup>6</sup>We use the notation  $\tilde{\mathbf{x}}$  for a “2D” point,  $\mathbf{x}$  for a 3D point, and  $\hat{\mathbf{x}}$  for a given or fixed 3D point. Likewise for line direction, plane normal, etc..

<sup>7</sup>If a pixel has the screen coordinate  $(u, v, 1)^T$ , its “2D” point on the panorama is represented by  $(u, v, f)^T$  where  $f$  is the focal length.

lems must be addressed in our modeling process.

## 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 the constraints above are bilinear. For example, a point on a plane is a bilinear constraint on both the point location and the plane normal. However, solving plane normals and line directions is independent of solving plane distance and points. Thus, in our system we decouple the modeling process into several linear steps:

- recovering camera orientations ( $\mathbf{R}$ ) from known line directions
- estimating plane normals ( $\mathbf{n}$ ) and line directions ( $\mathbf{m}$ )
- recovering camera translations ( $\mathbf{t}$ ) from known points
- estimating plane distances ( $d$ ), vertex positions ( $\mathbf{x}$ )

These steps are explained in detail in the following sections.

## 3 Recovering camera rotation

We discuss in this section how to recover camera orientations from known line directions. 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 assign them to be parallel to the  $Z$  axis of the world coordinate system.

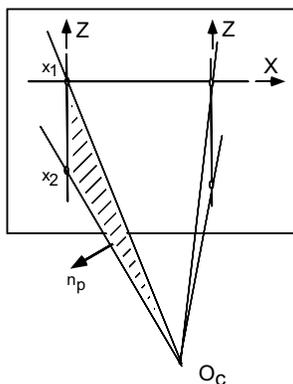


Figure 2: Camera rotation from 3 line directions

### 3.1 Minimum condition

**Lemma 1 (Minimum condition)** *Given two vertical lines and a horizontal line, the camera rotation matrix can be recovered.*

Each line forms a projection plane (with normal  $\tilde{\mathbf{n}}_p$ ) through the camera origin. Given two points  $\tilde{\mathbf{x}}_1$  and  $\tilde{\mathbf{x}}_2$  on a line, the plane normal can be computed by the cross product  $\tilde{\mathbf{n}}_p = \tilde{\mathbf{x}}_1 \times \tilde{\mathbf{x}}_2$  (Figure 2). The length of  $\tilde{\mathbf{n}}_p$  is a good confidence (or certainty) measure of the normal  $\tilde{\mathbf{n}}_p$ .

Let the camera rotation  $\mathbf{R} = [\mathbf{r}_x \mathbf{r}_y \mathbf{r}_z]$ . Each vertical line parallel to the  $Z$  axis (and the plane formed with the origin) gives a constraint  $\tilde{\mathbf{n}}_p \cdot \mathbf{r}_z = 0$ .

From two known vertical lines,  $\tilde{\mathbf{n}}_{p1} \cdot \mathbf{r}_z = 0$ , and  $\tilde{\mathbf{n}}_{p2} \cdot \mathbf{r}_z = 0$ , we have  $\mathbf{r}_z = \tilde{\mathbf{n}}_{p1} \times \tilde{\mathbf{n}}_{p2}$ . Note that there is sign ambiguity for the recovered  $\mathbf{r}_z$ . If we have more vertical lines, we can formulate the recovery of  $\mathbf{r}_z$  as a standard minimum eigenvalue problem.

With a known horizontal line (e.g., parallel to the  $X$  axis), we have a constraint on  $\mathbf{r}_x$  ( $\tilde{\mathbf{n}}_{pj} \cdot \mathbf{r}_x = 0$ ). Thus  $\mathbf{r}_x = \mathbf{r}_z \times \tilde{\mathbf{n}}_{pj}$  because  $\mathbf{r}_z \cdot \mathbf{r}_x = 0$ . Again there is sign ambiguity for the recovered  $\mathbf{r}_x$ . Finally  $\mathbf{r}_y = \mathbf{r}_z \times \mathbf{r}_x$  completes  $\mathbf{R}$ .

Obviously, the camera rotation can also be computed if two horizontal lines (e.g., parallel to  $X$  axis) and a vertical line are known.

### 3.2 Overconstrained problem

**Lemma 2 (General condition)** *Given at least two sets of parallel lines of known direc-*

tions, the camera rotation matrix can be recovered.

We now show how to incorporate all constraints to recover camera pose using unit quaternion. As shown above, if we have a pair of parallel lines with “line projection plane” normals as  $\tilde{\mathbf{n}}_{pj}$  and  $\tilde{\mathbf{n}}_{pk}$ , the line direction  $\tilde{\mathbf{m}}_i$  can be estimated as  $\tilde{\mathbf{n}}_{pj} \times \tilde{\mathbf{n}}_{pk}$ . Again, rather than normalizing  $\tilde{\mathbf{m}}_i$ , we can leave it unnormalized since its magnitude denotes own confidence in this measurement. Given the true line direction  $\hat{\mathbf{m}}_i$  in the world coordinate, we can formulate the camera pose recovery as

$$\arg \max_{\mathbf{R}} \sum_{i=1}^N \hat{\mathbf{m}}_i \cdot (\mathbf{R}^T \tilde{\mathbf{m}}_i) \quad (1)$$

with  $N \geq 2$ , which leads to a maximum eigenvalue problem using standard unit quaternion.

However, the resulting camera rotation  $\mathbf{R}$  can still be ambiguous due to the sign ambiguities in line directions  $\tilde{\mathbf{m}}$ . We solve this problem by specifying a few vertices with known coordinates, rather than asking the user to distinguish end points for each line, e.g., to differentiate (1,0,0) from (-1,0,0).

### 3.3 Line direction from parallel lines

More generally, 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

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

This is equivalent to finding the vanishing point of the lines [CW90]. 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.

## 4 Estimating plane normals

Once we have camera pose, we can recover the scene geometry (i.e., points, lines, and planes). Because of the bilinear constraints (such as points on planes), we recover plane normals ( $\mathbf{n}$ )

before solving plane distance ( $d$ ) and points ( $\mathbf{x}$ ). If the normal is given (N, S, up, down, etc.), it can be enforced as a hard constraint (see Section 6.1). 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 ( $\tilde{\mathbf{m}}$ ) 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  $\mathbf{n} = \mathbf{m}_1 \times \mathbf{m}_2$ .

A rectangle is a special case of parallelogram. We can recover the plane normal of a rectangle if we have 3 lines of the rectangle. Similar to parallelogram, we get one line direction  $\mathbf{m}_1$  from 2 parallel lines. And because of  $\mathbf{m}_1 \cdot \mathbf{m}_2 = 0$  and  $\mathbf{n}_{p2} \cdot \mathbf{m}_2 = 0$ , where  $\mathbf{n}_{p2} = \mathbf{R}^T \tilde{\mathbf{n}}_{p2}$ , we obtain the other line direction  $\mathbf{m}_2 = \mathbf{m}_1 \times \mathbf{n}_{p2}$ . Unlike [Har89], we do not need to know specifically the four corners of a rectangle.

Therefore, we can recover the surface orientation of an arbitrary plane (e.g., tilted ceiling) provided either we can draw a parallelogram (or rectangle) on the plane.

## 5 Recovering camera translation

A point on a 2D model (panorama) represents a ray from the camera origin through the pixel on the image. This constraint can be expressed in different ways. For example, we can relate each point in 3D model to its “2D” counterpart by a scale  $k$ , i.e.,

$$(\mathbf{x} - \mathbf{t}) = k\mathbf{R}^T \tilde{\mathbf{x}}. \quad (3)$$

Alternatively, the 3D point should lie on the ray represented by the 2D point,

$$(\mathbf{x} - \mathbf{t}) \times \mathbf{R}^T \tilde{\mathbf{x}} = 0, \quad (4)$$

which is equivalent to

$$(\mathbf{x} - \mathbf{t}) \cdot (\mathbf{R}^T \tilde{\mathbf{p}}_j) = 0, j = 0, 1, 2, \quad (5)$$

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.<sup>8</sup> Thus, camera translation  $\mathbf{t}$  can be recovered as a linear least-squares problem if we have two or more known 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 fixed points are also used to eliminate the ambiguities in recovering camera pose.

For a single panorama, the camera translation  $\mathbf{t}$  is set to zero if no point in 3D model is fixed, i.e., this implies that the camera coordinate coincides with the 3D model coordinate. We should point out that it is not necessary to recover camera translation independently; it can be solved for along with plane distance and points as shown in the next section.

## 6 Estimating the 3D model

### 6.1 Hard and soft constraints

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 differentiating 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 [GV96].

Some constraints (e.g., a point is known) are inherently hard, therefore equality constraints. Some constraints (e.g., a point on a “2D” model or panorama) are most appropriate as soft constraints because they are based on noisy image measurements. But most constraints can be considered as either hard or soft. It is a design decision why and when those constraints should be considered hard. Table 1 lists all constraints used in our modeling system. Again, we use the notations  $\hat{\mathbf{m}}$  and  $\hat{\mathbf{n}}$  to represent the known line direction  $\mathbf{m}$  and plane normal  $\mathbf{n}$ , respectively.

Take a point on a plane for an example. If the plane normal  $\hat{\mathbf{n}}_k$  is fixed, we consider the constraint  $(\mathbf{x}_i \cdot \hat{\mathbf{n}}_k + d_k = 0)$  as hard. 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,

---

<sup>8</sup>The third constraint with minimum  $\|\hat{\mathbf{p}}_i\|^2$  is eliminated.

Type	Constraint	n	Soft	Hard
Known point	$\hat{\mathbf{x}}_i$	3		x
Known plane	$\hat{d}_i$	1		x
planes	$d_i - d_j$	1		x
Point/model	$(\mathbf{x} - \mathbf{t}) \cdot \mathbf{p}_j = 0$	2	x	
Point/plane	$\mathbf{x}_i \cdot \hat{\mathbf{n}}_k + d_k = 0$	1		x
Point/plane	$\mathbf{x}_i \cdot \mathbf{n}_k + d_k = 0$	1	x	
Points/line	$(\mathbf{x}_i - \mathbf{x}_j) \times \hat{\mathbf{m}} = 0$	2		x
Points/line	$(\mathbf{x}_i - \mathbf{x}_j) \times \mathbf{m} = 0$	2	x	
known length	$\mathbf{x}_i - \mathbf{x}_j = c\hat{\mathbf{m}}$	3		x
known length	$\mathbf{x}_i - \mathbf{x}_j = c\mathbf{m}$	3	x	

Table 1: Hard and soft constraints (the third column represents the number of constraints)

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. One has to be cautious not to have too many hard constraints, which could conflict one another or make other soft constraints insignificant.

## 6.2 Solving equality-constrained L-S

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)

$$\mathbf{Ax} = \mathbf{b} \tag{6}$$

subject to (hard constraints)

$$\mathbf{Cx} = \mathbf{q} \tag{7}$$

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 [GV96]. Suppose  $\mathbf{C}$  is of full rank, let

$$\mathbf{C}^T = \mathbf{Q} \begin{bmatrix} \mathbf{R} \\ 0 \end{bmatrix} \quad (8)$$

be the QR factorization of  $\mathbf{C}^T$  where  $\mathbf{Q}$  ( $n \times n$ ) is orthogonal,  $\mathbf{Q}\mathbf{Q}^T = \mathbf{I}$ , and  $\mathbf{R}$  is  $p \times p$ .

If we define  $\mathbf{Q}^T \mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix}$ ,  $\mathbf{A}\mathbf{Q} = (\mathbf{A}_1, \mathbf{A}_2)$ , where  $\mathbf{A}_1$  is  $m \times p$ ,  $\mathbf{A}_2$  is  $m \times (n - p)$ ,  $\mathbf{x}_1$  is  $p \times 1$ , and  $\mathbf{x}_2$  is  $(n - p) \times 1$  we can solve  $\mathbf{x}_1$  because  $\mathbf{R}$  is upper triangular and

$$\begin{aligned} \mathbf{C}\mathbf{x} &= \mathbf{C}\mathbf{Q}\mathbf{Q}^T \mathbf{x} \\ &= \mathbf{P}\mathbf{R}^T \mathbf{x}_1 = \mathbf{q}. \end{aligned}$$

Then we solve  $\mathbf{x}_2$  from the unconstrained least-squares  $\|\mathbf{A}_2 \mathbf{x}_2 - (\mathbf{b} - \mathbf{A}_1 \mathbf{x}_1)\|^2$  because

$$\begin{aligned} \mathbf{A}\mathbf{x} - \mathbf{b} &= \mathbf{A}\mathbf{Q}\mathbf{Q}^T \mathbf{x} - \mathbf{b} \\ &= \mathbf{A}_1 \mathbf{x}_1 + \mathbf{A}_2 \mathbf{x}_2 - \mathbf{b} \\ &= \mathbf{A}_2 \mathbf{x}_2 - (\mathbf{b} - \mathbf{A}_1 \mathbf{x}_1). \end{aligned}$$

Finally  $\mathbf{x} = \mathbf{Q} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix}$ .

If  $\mathbf{C}$  is not of full rank, other approaches such as the elimination method [SS97a] can be used.

### 6.3 Decomposing the linear system

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 consists of several subsystems (connected components) which can be solved independently.<sup>9</sup> For example, when modeling a room with a computer monitor floating in the space not connected with any wall, ceiling or floor, we 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

---

<sup>9</sup>Ingnore the effect of camera translation.

- the number of equations<sup>10</sup> is no fewer than the number of unknowns,
- the right hand side is non-zero vector, i.e., some minimal ground truth data has been provided<sup>11</sup>
- the hard constraints are consistent.<sup>12</sup>

If any of the above conditions is not satisfied, the system is declared unsolvable, a warning message is then generated to indicate which set of unknowns cannot be recovered.

## 7 Multiple panoramas

To build 3D models from multiple panoramas, we do the following:

1. insert a new panorama
2. repeat same steps of modeling from a single panorama to obtain a rough camera pose for current view
3. project the existing model on the current panorama by the rough camera pose
4. fix up the predicated feature locations by dragging (or snapping) them to the right positions
5. recover a new model using multiple panoramas.

To recover the 3D model using multiple panoramas, we use bundle adjustment either update  $(d, \mathbf{x}, \mathbf{t})$  using linear least-squares, or also update  $(\mathbf{R}, \mathbf{m}, \mathbf{n})$  using full bilinear constrained non-linear least-squares. However, to bundle adjust different camera translations, it is more optimal [Zha97] to modify Eq.(5) such that

$$\frac{(\mathbf{x} - \mathbf{t})}{\|\mathbf{x} - \mathbf{t}\|} \cdot (\mathbf{R}^T \tilde{\mathbf{p}}_j) = 0, j = 0, 1. \quad (9)$$

---

<sup>10</sup>This includes both hard and soft constraints.

<sup>11</sup>In principle, we can still solve the system without any ground truth for multiple panoramas, but this requires finding the unit norm solution of the homogeneous set of equations.

<sup>12</sup>We can use QR decomposition of  $\mathbf{C}\mathbf{x} = \mathbf{q}$ , or  $\mathbf{Q}\mathbf{R}\mathbf{x} = \mathbf{q}$ , or  $\mathbf{R}\mathbf{x} = \mathbf{Q}^T \mathbf{q}$  to check if all zero rows of  $\mathbf{R}$  correspond to zero entries of  $\mathbf{Q}^T \mathbf{q}$ .



Figure 3: 3D model from a single panorama.



Figure 4: Texture maps (6) for the 3D model.

## 8 Experiments

We have implemented our system on a PC and tested with single and multiple panoramas. The system mainly consists of two parts: the interface (viewing the panorama with pan, tilt and zoom) and the modeler (recovering camera pose and the model). Figure 3 shows a spherical panorama 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 panorama [SS97a]. Corresponding (6) texture maps (without top and bottom faces) are shown in Figure 4. We observe that texture maps in Figure 4 have different sampling rates from original images. The sampling is the best (e.g., the image at upper right) when the surface normal is parallel with the viewing direction from the camera center, and the worst (e.g., the image at lower second from left) when



Figure 5: Two views of the interactive system.

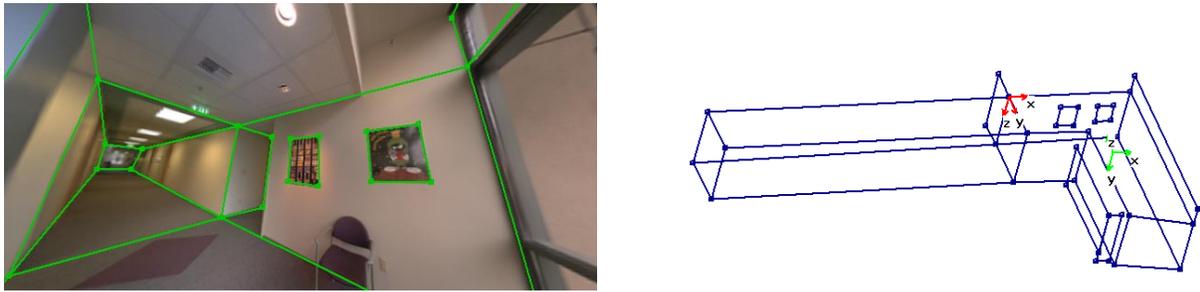


Figure 6: More complex 3D model from a single panorama.

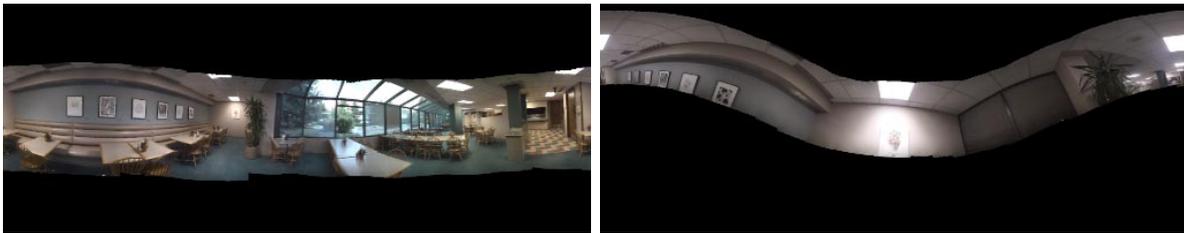


Figure 7: Two input panoramas of an indoor scene.

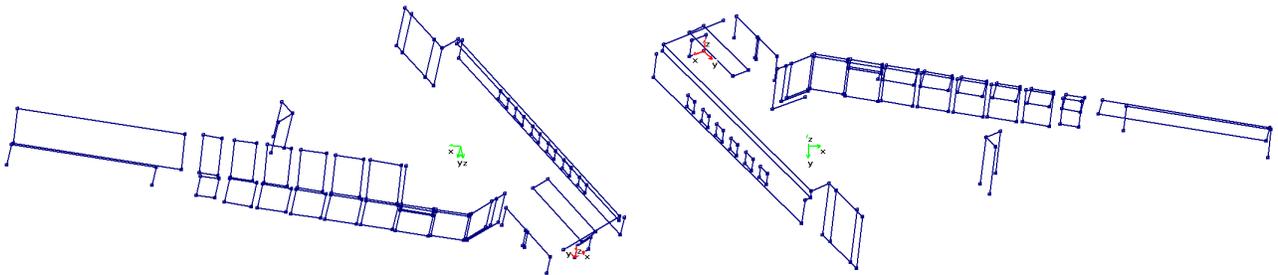


Figure 8: Two views of 3D wireframe model from multiple panorama.

perpendicular. This also explains why the sampling on the left is better than that on the right for the upper left image.

Figure 5 shows two views of system interaction. Green lines and points are the “2D” items that are manually drawn and assigned with properties, and blue lines and points are the recovered 3D model. The system is not difficult to use. It takes about 15 minutes for the authors to build the simple model in Figure 3. With extra effort, we can build a more complicated model (Figure 6) in 30 minutes.

We show another example of building 3D models from multiple panoramas. Figure 7 shows two spherical panoramas built from sequences of images taken with hand-held digital video camera. Figure 8a shows two views of reconstructed 3D wireframe model from the two panoramas in Figure 7. Notice that the occluded area (middle of the first panorama) by the first panorama is also recovered because it is visible in the second panorama.

## 9 Discussion and conclusions

Summary of the work. Why is it worthwhile? Improvement? Future?

In this paper, we have presented an interactive modeling system from a single or multiple panoramas. After constructing panoramas (each from a sequence of images taken around same viewpoint), our system first recovers the camera pose for each panorama from known line directions, and then constructs the 3D model using all possible constraints. In particular, we carefully divide the constraints into “hard” and “soft” so that the model processing becomes a linearly-constrained least-squares problem.

By decomposing the modeling process into a zero baseline problem (panorama construction) and a wide baseline problem, we avoid some ambiguities associated with traditional structure from motion approaches used for scene modeling. Our modeling system constructs accurate wire-frame and realistic texture-mapped 3D models. These results show that it is desirable and practical for the modeling system to take advantage of many regularities and priori knowledge existing in man-made environments (such as vertice lines and planes). The research problem is how to satisfy all the constraints in a consistent and coherent way.

Several design issues may need to be revisited later to improve the usability of our system.

For example, we have implemented the automatic line snapping which snaps the line to its closest edge present in the panorama. We may want to incorporate automatic line detection, corner detection and other feature detections to further automate the system. Robustification is another possibility.

We are also actively studying the problem of inverse texture mapping. In particular, we want to study what is the optimal way of extracting texture maps from multiple panoramas, given an accurate 3D model or a rough 3D model. Another direction of future reserach is to solve the multiple baseline stereo problem[KS96] using the initial model from the interative system.

## References

- [BB95] S. Becker and V. M. Bove. Semiautomatic 3-D model extraction from uncalibrated 2-d camera views. In *SPIE Vol. 2410, Visual Data Exploration and Analysis II*, pages 447–461, San Jose, CA, February 1995. Society of Photo-Optical Instrumentation Engineers.
- [BR97] 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.
- [CW90] R. T. Collins and R. S. Weiss. Vanish 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.
- [GV96] G. Golub and C. F. Van Loan. *Matrix Computation, third edition*. The John Hopkins University Press, Baltimore and London, 1996.
- [Har89] R. M. Haralick. Determining camera parameters from the perspective projection of a rectangle. *Pattern Recognition*, 22(3):225–230, 1989.

- [KS96] S. B. Kang and R. Szeliski. 3-D scene data recovery using omnidirectional multi-baseline stereo. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'96)*, pages 364–370, San Francisco, California, June 1996.
- [KW97] S. B. Kang and R. Weiss. Characterization of errors in compositing panoramic images. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'97)*, pages 103–109, San Juan, Puerto Rico, June 1997.
- [MB95] L. McMillan and G. Bishop. Plenoptic modeling: An image-based rendering system. *Computer Graphics (SIGGRAPH'95)*, pages 39–46, August 1995.
- [SK97] H. S. Sawhney and R. Kumar. True multi-image alignment and its application to mosaicing and lens distortion correction. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'97)*, pages 450–456, San Juan, Puerto Rico, June 1997.
- [SS97a] H.-Y. Shum and R. Szeliski. Panoramic image mosaicing. Technical Report MSR-TR-97-23, Microsoft Research, September 1997.
- [SS97b] R. Szeliski and H.-Y. Shum. Creating full view panoramic image mosaics and texture-mapped models. In *Computer Graphics Proceedings, Annual Conference Series*, pages 251–258, Proc. SIGGRAPH'97 (Los Angeles), August 1997. ACM SIGGRAPH.
- [Ste95] G. Stein. Accurate internal camera calibration using rotation, with analysis of sources of error. In *Fifth International Conference on Computer Vision (ICCV'95)*, pages 230–236, Cambridge, Massachusetts, June 1995.
- [TDM96] C. J. Taylor, P. E. Debevec, and J. Malik. Reconstructing polyhedral models of architectural scenes from photographs. In *Fourth European Conference on Computer Vision (ECCV'96)*, volume 2, pages 659–668, Cambridge, England, April 1996. Springer-Verlag.

- [Zha97] Z. Zhang. Determining the epipolar geometry and its uncertainty: A review. *International Journal of Computer Vision*, accepted 1997.