To appear in the SIGGRAPH conference proceedings

Modeling and Rendering Architecture from Photographs: A hybrid geometry- and image-based approach

Paul E. Debevec Camillo J. Taylor Jitendra Malik

University of California at Berkeley[1]

ABSTRACT

We present a new approach for modeling and rendering existing architectural scenes from a sparse set of still photographs. Our modeling approach, which combines both geometry-based and image-based techniques, has two components. The first component is a *photogrammetric modeling* method which facilitates the recovery of the basic geometry of the photographed scene. Our photogrammetric modeling approach is effective, convenient, and robust because it exploits the constraints that are characteristic of architectural scenes. The second component is a *model-based* stereo algorithm, which recovers how the real scene deviates from the basic model. By making use of the model, our stereo technique robustly recovers accurate depth from widely-spaced image pairs. Consequently, our approach can model large architectural environments with far fewer photographs than current image-based modeling approaches. For producing renderings, we present *view-dependent texture mapping*, a method of compositing multiple views of a scene that better simulates geometric detail on basic models. Our approach can be used to recover models for use in either geometry-based or image-based rendering systems. We present results that demonstrate our approach's ability to create realistic renderings of architectural scenes from viewpoints far from the original photographs.

CR Descriptors: I.2.10 [**Artificial Intelligence**]: Vision and Scene Understanding - *Modeling and recovery of physical attributes*; I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism - *Color, shading, shadowing, and texture* I.4.8 [**Image Processing**]: Scene Analysis - *Stereo*; J.6 [**Computer-Aided Engineering**]: Computer-aided design (CAD).

1 INTRODUCTION

Efforts to model the appearance and dynamics of the real world have produced some of the most compelling imagery in computer graphics. In particular, efforts to model architectural scenes, from the Amiens Cathedral to the Giza Pyramids to Berkeley's Soda Hall, have produced impressive walk-throughs and inspiring flybys. Clearly, it is an attractive application to be able to explore the world's architecture unencumbered by fences, gravity, customs, or jetlag.

[1]Computer Science Division, University of California at Berkeley, Berkeley, CA 94720-1776. {debevec,camillo,malik}@cs.berkeley.edu. See also http://www.cs.berkeley.edu/~debevec/Research

Unfortunately, current geometry-based methods (Fig. 1a) of modeling existing architecture, in which a modeling program is used to manually position the elements of the scene, have several drawbacks. First, the process is extremely labor-intensive, typically involving surveying the site, locating and digitizing architectural plans (if available), or converting existing CAD data (again, if available). Second, it is difficult to verify whether the resulting model is accurate. Most disappointing, though, is that the renderings of the resulting models are noticeably computer-generated; even those that employ liberal texture-mapping generally fail to resemble real photographs.

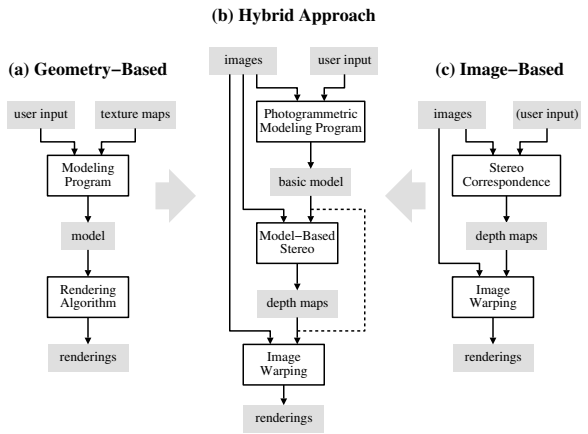


Figure 1: *Schematic of how our hybrid approach combines geometry-based and image-based approaches to modeling and rendering architecture from photographs.*

Recently, creating models directly from photographs has received increased interest in computer graphics. Since real images are used as input, such an image-based system (Fig. 1c) has an advantage in producing photorealistic renderings as output. Some of the most promising of these systems [16, 13] rely on the computer vision technique of computational stereopsis to automatically determine the structure of the scene from the multiple photographs available. As a consequence, however, these systems are only as strong as the underlying stereo algorithms. This has caused problems because state-of-the-art stereo algorithms have a number of significant weaknesses; in particular, the photographs need to appear very similar for reliable results to be obtained. Because of this, current image-based techniques must use many closely spaced images, and in some cases employ significant amounts of user input for each image pair to supervise the stereo algorithm. In this framework, capturing the data for a realistically renderable model would require an impractical number of closely spaced photographs, and deriving the depth from the photographs could require an impractical amount of user input. These concessions to the weakness of stereo algorithms bode poorly for creating large-scale, freely navigable virtual environments from photographs.

Our research aims to make the process of modeling architectural

scenes more convenient, more accurate, and more photorealistic than the methods currently available. To do this, we have developed a new approach that draws on the strengths of both geometry-based and image-based methods, as illustrated in Fig. 1b. The result is that our approach to modeling and rendering architecture requires only a sparse set of photographs and can produce realistic renderings from arbitrary viewpoints. In our approach, a basic geometric model of the architecture is recovered interactively with an easy-to-use photogrammetric modeling system, novel views are created using view-dependent texture mapping, and additional geometric detail can be recovered automatically through stereo correspondence. The final images can be rendered with current image-based rendering techniques. Because only photographs are required, our approach to modeling architecture is neither invasive nor does it require architectural plans, CAD models, or specialized instrumentation such as surveying equipment, GPS sensors or range scanners.

1.1 Background and Related Work

The process of recovering 3D structure from 2D images has been a central endeavor within computer vision, and the process of rendering such recovered structures is a subject receiving increased interest in computer graphics. Although no general technique exists to derive models from images, four particular areas of research have provided results that are applicable to the problem of modeling and rendering architectural scenes. They are: Camera Calibration, Structure from Motion, Stereo Correspondence, and Image-Based Rendering.

1.1.1 Camera Calibration

Recovering 3D structure from images becomes a simpler problem when the cameras used are *calibrated*, that is, the mapping between image coordinates and directions relative to each camera is known. This mapping is determined by, among other parameters, the camera's focal length and its pattern of radial distortion. Camera calibration is a well-studied problem both in photogrammetry and computer vision; some successful methods include [20] and [5]. While there has been recent progress in the use of uncalibrated views for 3D reconstruction [7], we have found camera calibration to be a straightforward process that considerably simplifies the problem.

1.1.2 Structure from Motion

Given the 2D projection of a point in the world, its position in 3D space could be anywhere on a ray extending out in a particular direction from the camera's optical center. However, when the projections of a sufficient number of points in the world are observed in multiple images from different positions, it is theoretically possible to deduce the 3D locations of the points as well as the positions of the original cameras, up to an unknown factor of scale.

This problem has been studied in the area of photogrammetry for the principal purpose of producing topographic maps. In 1913, Kruppa [10] proved the fundamental result that given two views of five distinct points, one could recover the rotation and translation between the two camera positions as well as the 3D locations of the points (up to a scale factor). Since then, the problem's mathematical and algorithmic aspects have been explored starting from the fundamental work of Ullman [21] and Longuet-Higgins [11], in the early 1980s. Faugeras's book [6] overviews the state of the art as of 1992. So far, a key realization has been that the recovery of structure is very sensitive to noise in image measurements when the translation between the available camera positions is small.

Attention has turned to using more than two views with image stream methods such as [19] or recursive approaches (e.g. [1]). [19] shows excellent results for the case of orthographic cameras, but direct solutions for the perspective case remain elusive. In general, linear algorithms for the problem fail to make use of all available information while nonlinear minimization methods are prone to difficulties arising from local minima in the parameter space. An alternative formulation of the problem [17] uses lines rather than points as image measurements, but the previously stated concerns were shown to remain largely valid. For purposes of computer graphics, there is yet another problem: the models recovered by these algorithms consist of sparse point fields or individual line segments, which are not directly renderable as solid 3D models.

In our approach, we exploit the fact that we are trying to recover geometric models of architectural scenes, not arbitrary three-dimensional point sets. This enables us to include additional constraints not typically available to structure from motion algorithms and to overcome the problems of numerical instability that plague such approaches. Our approach is demonstrated in a useful interactive system for building architectural models from photographs.

1.1.3 Stereo Correspondence

The geometrical theory of structure from motion assumes that one is able to solve the *correspondence* problem, which is to identify the points in two or more images that are projections of the same point in the world. In humans, corresponding points in the two slightly differing images on the retinas are determined by the visual cortex in the process called binocular stereopsis.

Years of research (e.g. [2, 4, 8, 9, 12, 15]) have shown that determining stereo correspondences by computer is difficult problem. In general, current methods are successful only when the images are similar in appearance, as in the case of human vision, which is usually obtained by using cameras that are closely spaced relative to the objects in the scene. When the distance between the cameras (often called the *baseline*) becomes large, surfaces in the images exhibit different degrees of foreshortening, different patterns of occlusion, and large disparities in their locations in the two images, all of which makes it much more difficult for the computer to determine correct stereo correspondences. Unfortunately, the alternative of improving stereo correspondence by using images taken from nearby locations has the disadvantage that computing depth becomes very sensitive to noise in image measurements.

In this paper, we show that having an approximate model of the photographed scene makes it possible to robustly determine stereo correspondences from images taken from widely varying viewpoints. Specifically, the model enables us to warp the images to eliminate unequal foreshortening and to predict major instances of occlusion *before* trying to find correspondences.

1.1.4 Image-Based Rendering

In an image-based rendering system, the model consists of a set of images of a scene and their corresponding depth maps. When the depth of every point in an image is known, the image can be re-rendered from any nearby point of view by projecting the pixels of the image to their proper 3D locations and reprojecting them onto a new image plane. Thus, a new image of the scene is created by warping the images according to their depth maps. A principal attraction of image-based rendering is that it offers a method of rendering arbitrarily complex scenes with a constant amount of computation required per pixel. Using this property, [23] demonstrated how regularly spaced synthetic images (with their computed depth maps) could be warped and composited in real time to produce a virtual environment.

More recently, [13] presented a real-time image-based rendering system that used panoramic photographs with depth computed, in part, from stereo correspondence. One finding of the paper was that extracting reliable depth estimates from stereo is "very difficult". The method was nonetheless able to obtain acceptable results for nearby views using user input to aid the stereo depth recovery: the correspondence map for each image pair was seeded with 100 to 500 user-supplied point correspondences and also post-processed. Even

with user assistance, the images used still had to be closely spaced; the largest baseline described in the paper was five feet.

The requirement that samples be close together is a serious limitation to generating a freely navigable virtual environment. Covering the size of just one city block would require thousands of panoramic images spaced five feet apart. Clearly, acquiring so many photographs is impractical. Moreover, even a dense lattice of ground-based photographs would only allow renderings to be generated from within a few feet of the original camera level, precluding any virtual fly-bys of the scene. Extending the dense lattice of photographs into three dimensions would clearly make the acquisition process even more difficult. The approach described in this paper takes advantage of the structure in architectural scenes so that it requires only a sparse set of photographs. For example, our approach has yielded a virtual fly-around of a building from just twelve standard photographs.

1.2 Overview

In this paper we present three new modeling and rendering techniques: photogrammetric modeling, view-dependent texture mapping, and model-based stereo. We show how these techniques can be used in conjunction to yield a convenient, accurate, and photorealistic method of modeling and rendering architecture from photographs. In our approach, the photogrammetric modeling program is used to create a basic volumetric model of the scene, which is then used to constrain stereo matching. Our rendering method composites information from multiple images with view-dependent texture-mapping. Our approach is successful because it splits the task of modeling from images into tasks which are easily accomplished by a person (but not a computer algorithm), and tasks which are easily performed by a computer algorithm (but not a person).

In Section 2, we present our **photogrammetric modeling** method. In essence, we have recast the structure from motion problem not as the recovery of individual point coordinates, but as the recovery of the parameters of a constrained hierarchy of parametric primitives. The result is that accurate architectural models can be recovered robustly from just a few photographs and with a minimal number of user-supplied correspondences.

In Section 3, we present **view-dependent texture mapping**, and show how it can be used to realistically render the recovered model. Unlike traditional texture-mapping, in which a single static image is used to color in each face of the model, view-dependent texture mapping interpolates between the available photographs of the scene depending on the user's point of view. This results in more lifelike animations that better capture surface specularities and unmodeled geometric detail.

Lastly, in Section 4, we present **model-based stereo**, which is used to automatically refine a basic model of a photographed scene. This technique can be used to recover the structure of architectural ornamentation that would be difficult to recover with photogrammetric modeling. In particular, we show that projecting pairs of images onto an initial approximate model allows conventional stereo techniques to robustly recover very accurate depth measurements from images with widely varying viewpoints.

2 Photogrammetric Modeling

In this section we present our method for photogrammetric modeling, in which the computer determines the parameters of a hierarchical model of parametric polyhedral primitives to reconstruct the architectural scene. We have implemented this method in *Façade*, an easy-to-use interactive modeling program that allows the user to construct a geometric model of a scene from digitized photographs. We first overview Façade from the point of view of the user, then we describe our model representation, and then we explain our reconstruction algorithm. Lastly, we present results from using Façade to reconstruct several architectural scenes.

2.1 The User's View

Constructing a geometric model of an architectural scene using Façade is an incremental and straightforward process. Typically, the user selects a small number of photographs to begin with, and models the scene one piece at a time. The user may refine the model and include more images in the project until the model meets the desired level of detail.

Fig. 2(a) and (b) shows the two types of windows used in Façade: image viewers and model viewers. The user instantiates the components of the model, marks edges in the images, and corresponds the edges in the images to the edges in the model. When instructed, Façade computes the sizes and relative positions of the model components that best fit the edges marked in the photographs.

Components of the model, called *blocks*, are parameterized geometric primitives such as boxes, prisms, and surfaces of revolution. A box, for example, is parameterized by its length, width, and height. The user models the scene as a collection of such blocks, creating new block classes as desired. Of course, the user does not need to specify numerical values for the blocks' parameters, since these are recovered by the program.

The user may choose to constrain the sizes and positions of any of the blocks. In Fig. 2(b), most of the blocks have been constrained to have equal length and width. Additionally, the four pinnacles have been constrained to have the same shape. Blocks may also be placed in constrained relations to one other. For example, many of the blocks in Fig. 2(b) have been constrained to sit centered and on top of the block below. Such constraints are specified using a graphical 3D interface. When such constraints are provided, they are used to simplify the reconstruction problem.

The user marks edge features in the images using a point-and-click interface; a gradient-based technique as in [14] can be used to align the edges with sub-pixel accuracy. We use edge rather than point features since they are easier to localize and less likely to be completely obscured. Only a section of each edge needs to be marked, making it possible to use partially visible edges. For each marked edge, the user also indicates the corresponding edge in the model. Generally, accurate reconstructions are obtained if there are as many correspondences in the images as there are free camera and model parameters. Thus, Façade reconstructs scenes accurately even when just a portion of the visible edges and marked in the images, and when just a portion of the model edges are given correspondences.

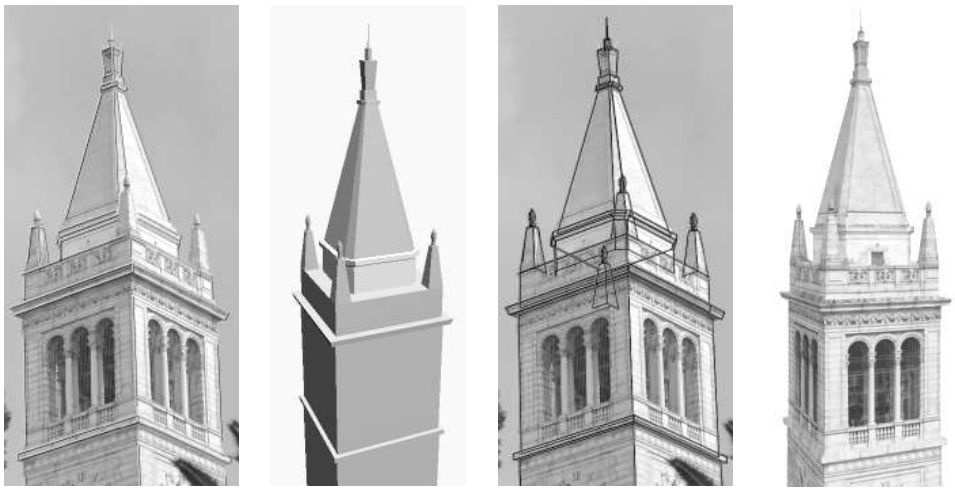
At any time, the user may instruct the computer to reconstruct the scene. The computer then solves for the parameters of the model that cause it to align with the marked features in the images. During the reconstruction, the computer computes and displays the locations from which the photographs were taken. For simple models consisting of just a few blocks, a full reconstruction takes only a few seconds; for more complex models, it can take a few minutes. For this reason, the user can instruct the computer to employ faster but less precise reconstruction algorithms (see Sec. 2.4) during the intermediate stages of modeling.

To verify the the accuracy of the recovered model and camera positions, Façade can project the model into the original photographs. Typically, the projected model deviates from the photographs by less than a pixel. Fig. 2(c) shows the results of projecting the edges of the model in Fig. 2(b) into the original photograph.

Lastly, the user may generate novel views of the model by positioning a virtual camera at any desired location. Façade will then use the view-dependent texture-mapping method of Section 3 to render a novel view of the scene from the desired location. Fig. 2(d) shows an aerial rendering of the tower model.

2.2 Model Representation

The purpose of our choice of model representation is to represent the scene as a surface model with as few parameters as possible: when



(a) (b) (c) (d)

Figure 2: **(a)** *A photograph of the Campanile, Berkeley's clock tower, with marked edges shown in green.* **(b)** *The model recovered by our photogrammetric modeling method. Although only the left pinnacle was marked, the remaining three (including one not visible) were recovered from symmetrical constraints in the model. Our method allows any number of images to be used, but in this case constraints of symmetry made it possible to recover an accurate 3D model from a single photograph.* **(c)** *The accuracy of the model is verified by reprojecting it into the original photograph through the recovered camera position.* **(d)** *A synthetic view of the Campanile generated using the view-dependent texture-mapping method described in Section 3. A real photograph from this position would be difficult to take since the camera position is 250 feet above the ground.*

the model has fewer parameters, the user needs to specify fewer correspondences, and the computer can reconstruct the model more efficiently. In Façade, the scene is represented as a constrained hierarchical model of parametric polyhedral primitives, called *blocks*. Each block has a small set of parameters which serve to define its size and shape. Each coordinate of each vertex of the block is then expressed as linear combination of the block's parameters, relative to an internal coordinate frame. For example, for the wedge block in Fig. 3, the coordinates of the vertex P_o are written in terms of the block parameters $width$, $height$, and $length$ as $P_o = (-width, -height, length)^T$. Each block is also given an associated bounding box.

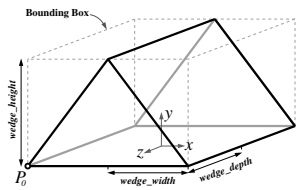
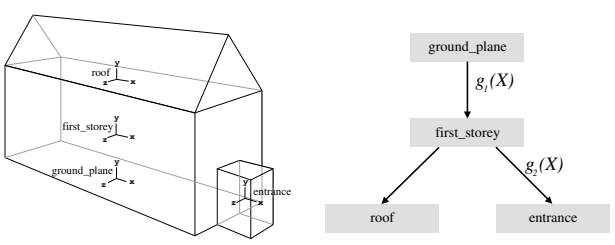


Figure 3: *A wedge block with its parameters and bounding box.*



(a) (b)

Figure 4: **(a)** *A geometric model of a simple building.* **(b)** *The model's hierarchical representation. The nodes in the tree represent parametric primitives (called* blocks*) while the links contain the spatial relationships between the blocks.*

The blocks in Façade are organized in a hierarchical tree structure as shown in Fig. 4(b). Each node of the tree represents an individual block, while the links in the tree contain the spatial relationships between blocks, called *relations*. Such hierarchical structures are also used in traditional modeling systems.

The relation between a block and its parent is most generally represented as a rotation matrix R and a translation vector t. This representation requires six parameters: three each for R and t. In architectural scenes, however, the relationship between two blocks usually has a simple form that can be represented with fewer parameters, and Façade allows the user to build such constraints on R and t into the model. The rotation R between a block and its parent can be specified in one of three ways: first, as an unconstrained rotation, requiring three parameters; second, as a rotation about a particular coordinate axis, requiring just one parameter; or third, as a fixed or null rotation, requiring no parameters.

Likewise, Façade allows for constraints to be placed on each component of the translation vector t. Specifically, the user can constrain the bounding boxes of two blocks to align themselves in some manner along each dimension. For example, in order to ensure that the roof block in Fig. 4 lies on top of the first story block, the user can require that the maximum y extent of the first story block be equal to the minimum y extent of the roof block. With this constraint, the translation along the y axis is computed ($t_y = (first_story_y^{MAX} - roof_y^{MIN})$) rather than represented as a parameter of the model.

Each parameter of each instantiated block is actually a reference to a named symbolic variable, as illustrated in Fig. 5. As a result, two parameters of different blocks (or of the same block) can be equated by having each parameter reference the same symbol. This facility allows the user to equate two or more of the dimensions in a model, which makes modeling symmetrical blocks and repeated structure more convenient. Importantly, these constraints reduce the number of degrees of freedom in the model, which, as we will show, simplifies the structure recovery problem.

Once the blocks and their relations have been parameterized, it is straightforward to derive expressions for the world coordinates of the block vertices. Consider the set of edges which link a specific block in the model to the ground plane as shown in Fig. 4.

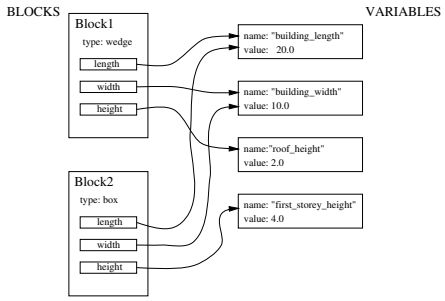


Figure 5: *Representation of block parameters as symbol references. A single variable can be referenced by the model in multiple places, allowing constraints of symmetry to be embedded in the model.*

Let $g_1(X), ..., g_n(X)$ represent the rigid transformations associated with each of these links, where X represents the vector of all the model parameters. The world coordinates $P_w(X)$ of a particular block vertex $P(X)$ is then:

$$P_w(X) = g_1(X)...g_n(X)P(X) \tag{1}$$

Similarly, the world orientation $v_w(X)$ of a particular line segment $v(X)$ is:

$$v_w(X) = g_1(X)...g_n(X)v(X) \tag{2}$$

In these equations, the point vectors P and P_w and the orientation vectors v and v_w are represented in homogeneous coordinates.

Modeling the scene with polyhedral blocks, as opposed to points, line segments, surface patches, or polygons, is advantageous for a number of reasons:

- Most architectural scenes are well modeled by an arrangement of geometric primitives.
- Blocks implicitly contain common architectural elements such as parallel lines and right angles.
- Manipulating block primitives is convenient since they are at a suitably high level of abstraction; individual features such as points and lines are less manageable.
- A surface model of the scene is readily obtained from the blocks, so there is no need to infer surfaces from discrete features.
- Modeling in terms of blocks and relationships greatly reduces the number of parameters that the reconstruction algorithm needs to recover.

The last point is crucial to the robustness of our reconstruction algorithm and the viability of our modeling system, and is illustrated best with an example. The model in Fig. 2 is parameterized by just 33 variables (the unknown camera position adds six more). If each block in the scene were unconstrained (in its dimensions and position), the model would have 240 parameters; if each line segment in the scene were treated independently, the model would have 2,896 parameters. This reduction in the number of parameters greatly enhances the robustness and efficiency of the method as compared to traditional structure from motion algorithms. Lastly, since the number of correspondences needed to suitably overconstrain the minimization is roughly proportional to the number of parameters in the model, this reduction means that the number of correspondences required of the user is manageable.

2.3 Reconstruction Algorithm

Our reconstruction algorithm works by minimizing an objective function $\mathcal{O}$ that sums the disparity between the projected edges of the model and the edges marked in the images, i.e. $\mathcal{O} = \sum Err_i$ where Err_i represents the disparity computed for edge feature i. Thus, the unknown model parameters and camera positions are computed by minimizing $\mathcal{O}$ with respect to these variables. Our system uses the the error function Err_i from [17], described below.

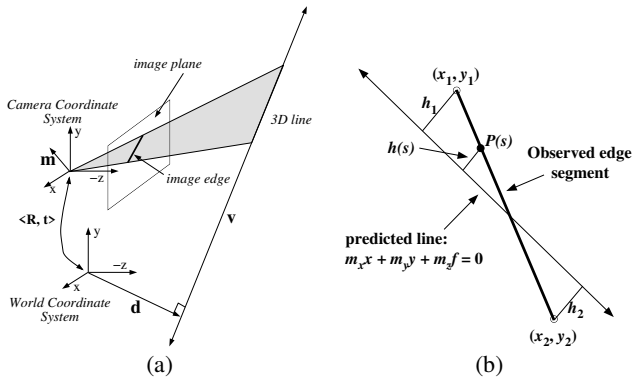


Figure 6: **(a)** *Projection of a straight line onto a camera's image plane.* **(b)** *The error function used in the reconstruction algorithm. The heavy line represents the observed edge segment (marked by the user) and the lighter line represents the model edge predicted by the current camera and model parameters.*

Fig. 6(a) shows how a straight line in the model projects onto the image plane of a camera. The straight line can be defined by a pair of vectors $\langle v, d \rangle$ where v represents the direction of the line and d represents a point on the line. These vectors can be computed from equations 2 and 1 respectively. The position of the camera with respect to world coordinates is given in terms of a rotation matrix R_j and a translation vector t_j. The normal vector denoted by $\mathbf{m}$ in the figure is computed from the following expression:

$$\mathbf{m} = R_j(\mathbf{v} \times (\mathbf{d} - t_j)) \tag{3}$$

The projection of the line onto the image plane is simply the intersection of the plane defined by m with the image plane, located at $z = -f$ where f is the focal length of the camera. Thus, the image edge is defined by the equation $m_x x + m_y y - m_z f = 0$.

Fig. 6(b) shows how the error between the observed image edge $\{(x_1, y_1), (x_2, y_2)\}$ and the predicted image line is calculated for each correspondence. Points on the observed edge segment can be parameterized by a single scalar variable $s \in [0, l]$ where l is the length of the edge. We let $h(s)$ be the function that returns the shortest distance from a point on the segment, $p(s)$, to the predicted edge.

With these definitions, the total error between the observed edge segment and the predicted edge is calculated as:

$$Err_i = \int_0^l h^2(s)ds = \frac{l}{3}(h_1^2 + h_1 h_2 + h_2^2) = \mathbf{m}^T(A^T B A)\mathbf{m} \tag{4}$$

where:

$$\begin{aligned} \mathbf{m} &= (m_x, m_y, m_z)^T \\ A &= \begin{pmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \end{pmatrix} \\ B &= \frac{l}{3(m_x^2 + m_y^2)} \begin{pmatrix} 1 & 0.5 \\ 0.5 & 1 \end{pmatrix} \end{aligned}$$

The final objective function $\mathcal{O}$ is the sum of the error terms resulting from each correspondence. We minimize $\mathcal{O}$ using a variant of the Newton-Raphson method, which involves calculating the gradient and Hessian of $\mathcal{O}$ with respect to the parameters of the camera

and the model. As we have shown, it is simple to construct symbolic expressions for m in terms of the unknown model parameters. The minimization algorithm differentiates these expressions symbolically to evaluate the gradient and Hessian after each iteration. The procedure is inexpensive since the expressions for d and v in Equations 2 and 1 have a particularly simple form.

2.4 Computing an Initial Estimate

The objective function described in Section 2.3 section is non-linear with respect to the model and camera parameters and consequently can have local minima. If the algorithm begins at a random location in the parameter space, it stands little chance of converging to the correct solution. To overcome this problem we have developed a method to directly compute a good initial estimate for the model parameters and camera positions that is near the correct solution. In practice, our initial estimate method consistently enables the non-linear minimization algorithm to converge to the correct solution.

Our initial estimate method consists of two procedures performed in sequence. The first procedure estimates the camera rotations while the second estimates the camera translations and the parameters of the model. Both initial estimate procedures are based upon an examination of Equation 3. From this equation the following constraints can be deduced:

$$m^T R_j \mathbf{v} = 0 \quad (5)$$

$$m^T R_j (\mathbf{d} - t_j) = 0 \quad (6)$$

Given an observed edge $\mathbf{u}_{ij}$ the measured normal $\mathbf{m}'$ to the plane passing through the camera center is:

$$\mathbf{m}' = \begin{pmatrix} x_1 \\ y_1 \\ -f \end{pmatrix} \times \begin{pmatrix} x_2 \\ y_2 \\ -f \end{pmatrix} \quad (7)$$

From these equations, we see that any model edges of known orientation constrain the possible values for R_j. Since most architectural models contain many such edges (e.g. horizontal and vertical lines), each camera rotation can be usually be estimated from the model independent of the model parameters and independent of the camera's location in space. Our method does this by minimizing the following objective function $\mathcal{O}_1$ that sums the extents to which the rotations R_j violate the constraints arising from Equation 5:

$$\mathcal{O}_1 = \sum_i (m^T R_j \mathbf{v}_i)^2, \quad \mathbf{v}_i \in \{\hat{x}, \hat{y}, \hat{z}\} \quad (8)$$

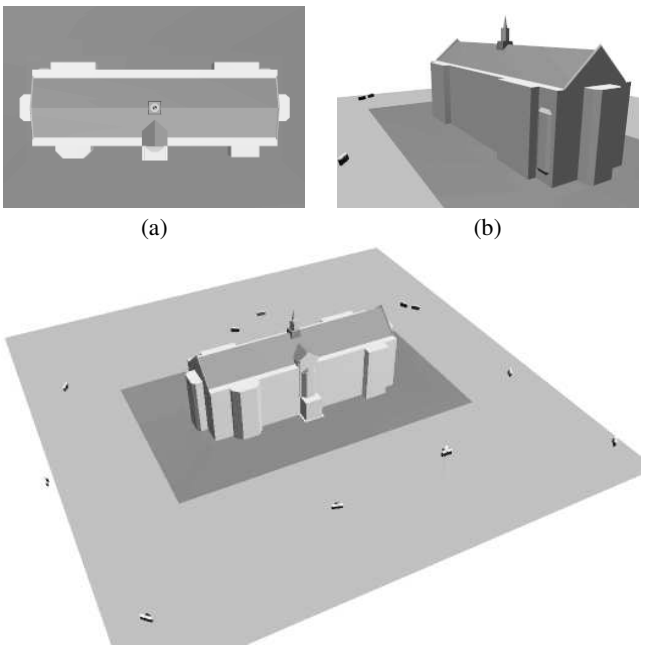
Once initial estimates for the camera rotations are computed, Equation 6 is used to obtain initial estimates of the model parameters and camera locations. Equation 6 reflects the constraint that all of the points on the line defined by the tuple $\langle \mathbf{v}, \mathbf{d} \rangle$ should lie on the plane with normal vector $\mathbf{m}$ passing through the camera center. This constraint is expressed in the following objective function $\mathcal{O}_2$ where $P_i(X)$ and $Q_i(X)$ are expressions for the vertices of an edge of the model.

$$\mathcal{O}_2 = \sum_i (m^T R_j (P_i(X) - t_j))^2 + (m^T R_j (Q_i(X) - t_j))^2 \quad (9)$$

In the special case where all of the block relations in the model have a known rotation, this objective function becomes a simple quadratic form which is easily minimized by solving a set of linear equations.

Once the initial estimate is obtained, the non-linear minimization over the entire parameter space is applied to produce the best possible reconstruction. Typically, the minimization requires fewer than ten iterations and adjusts the parameters of the model by at most a few percent from the initial estimates. The edges of the recovered models typically conform to the original photographs to within a pixel.

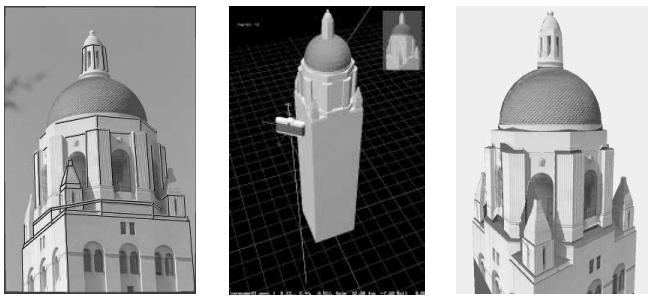
Figure 7: *Three of twelve photographs used to reconstruct the entire exterior of University High School in Urbana, Illinois. The superimposed lines indicate the edges the user has marked.*



(a) (b) (c)

Figure 8: *The high school model, reconstructed from twelve photographs.* **(a)** *Overhead view.* **(b)** *Rear view.* **(c)** *Aerial view showing the recovered camera positions. Two nearly coincident cameras can be observed in front of the building; their photographs were taken from the second story of a building across the street.*

Figure 9: *A synthetic view of University High School. This is a frame from an animation of flying around the entire building.*



(a) (b) (c)

Figure 10: *Reconstruction of Hoover Tower, Stanford, CA* **(a)** *Original photograph, with marked edges indicated.* **(b)** *Model recovered from the single photograph shown in (a).* **(c)** *Texture-mapped aerial view from the virtual camera position indicated in (b). Regions not seen in (a) are indicated in blue.*

2.5 Results

Fig. 2 showed the results of using Façade to reconstruct a clock tower from a single image. Figs. 7 and 8 show the results of using Façade to reconstruct a high school building from twelve photographs. (The model was originally constructed from just five images; the remaining images were added to the project for purposes of generating renderings using the techniques of Section 3.) The photographs were taken with a calibrated 35mm still camera with a standard 50mm lens and digitized with the PhotoCD process. Images at the 1536×1024 pixel resolution were processed to correct for lens distortion, then filtered down to 768×512 pixels for use in the modeling system. Fig. 8 shows some views of the recovered model and camera positions, and Fig. 9 shows a synthetic view of the building generated by the technique in Sec. 3.

Fig. 10 shows the reconstruction of another tower from a single photograph. The dome was modeled specially since the reconstruction algorithm does not recover curved surfaces. The user constrained a two-parameter hemisphere block to sit centered on top of the tower, and manually adjusted its height and width to align with the photograph. Each of the models presented took approximately four hours to create.

3 View-Dependent Texture-Mapping

In this section we present view-dependent texture-mapping, an effective method of rendering the scene that involves projecting the original photographs onto the model. This form of texture-mapping is most effective when the model conforms closely to the actual structure of the scene, and when the original photographs show the scene in similar lighting conditions. In Section 4 we will show how view-dependent texture-mapping can be used in conjunction with model-based stereo to produce realistic renderings when the recovered model only approximately models the structure of the scene.

Since the camera positions of the original photographs are recovered during the modeling phase, projecting the images onto the model is straightforward. In this section we first describe how we project a single image onto the model, and then how we merge several image projections to render the entire model. Unlike traditional texture-mapping, our method projects different images onto the model depending on the user's viewpoint. As a result, our view-dependent texture mapping can give a better illusion of additional geometric detail in the model.

3.1 Projecting a Single Image

The process of texture-mapping a single image onto the model can be thought of as replacing each camera with a slide projector that projects the original image onto the model. When the model is not convex, it is possible that some parts of the model will shadow others with respect to the camera. While such shadowed regions could be determined using an object-space visible surface algorithm, or an image-space ray casting algorithm, we use an image-space shadow map algorithm based on [22] since it is efficiently implemented using z-buffer hardware.

Fig. 11, upper left, shows the results of mapping a single image onto the high school building model. The recovered camera position for the projected image is indicated in the lower left corner of the image. Because of self-shadowing, not every point on the model within the camera's viewing frustum is mapped.

3.2 Compositing Multiple Images

In general, each photograph will view only a piece of the model. Thus, it is usually necessary to use multiple images in order to render the entire model from a novel point of view. The top images of Fig. 11 show two different images mapped onto the model and rendered from a novel viewpoint. Some pixels are colored in just one of the renderings, while some are colored in both. These two renderings can be merged into a composite rendering by considering the corresponding pixels in the rendered views. If a pixel is mapped in only one rendering, its value from that rendering is used in the composite. If it is mapped in more than one rendering, the renderer has to decide which image (or combination of images) to use.

It would be convenient, of course, if the projected images would agree perfectly where they overlap. However, the images will not necessarily agree if there is unmodeled geometric detail in the building, or if the surfaces of the building exhibit non-Lambertian reflection. In this case, the best image to use is clearly the one with the viewing angle closest to that of the rendered view. However, using the image closest in angle at every pixel means that neighboring rendered pixels may be sampled from different original images. When this happens, specularity and unmodeled geometric detail can cause visible seams in the rendering. To avoid this problem, we smooth these transitions through weighted averaging as in Fig. 12.

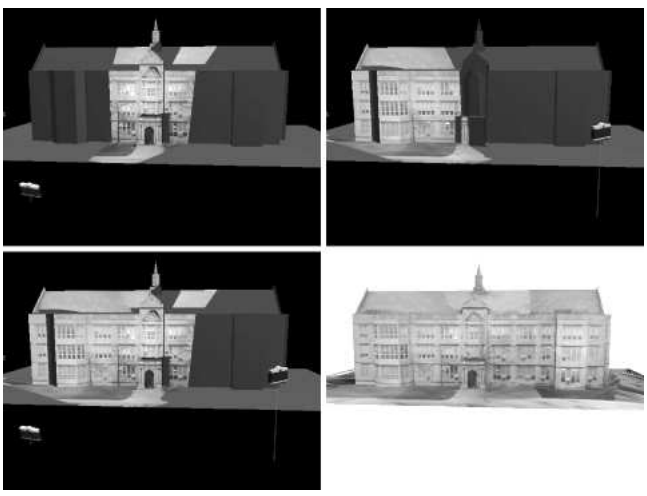


Figure 11: *The process of assembling projected images to form a composite rendering. The top two pictures show two images projected onto the model from their respective recovered camera positions. The lower left picture shows the results of compositing these two renderings using our view-dependent weighting function. The lower right picture shows the results of compositing renderings of all twelve original images. Some pixels near the front edge of the roof not seen in any image have been filled in with the hole-filling algorithm from [23].*

Even with this weighting, neighboring pixels can still be sampled from different views at the boundary of a projected image, since the contribution of an image must be zero outside its boundary. To

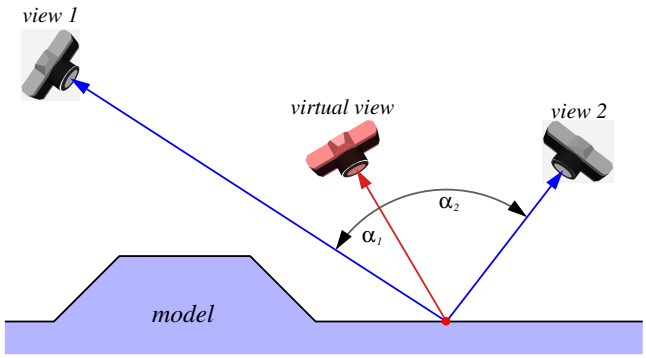


Figure 12: *The weighting function used in view-dependent texture mapping. The pixel in the virtual view corresponding to the point on the model is assigned a weighted average of the corresponding pixels in actual views 1 and 2. The weights w_1 and w_2 are inversely inversely proportional to the magnitude of angles α_1 and α_2. Alternately, more sophisticated weighting functions based on expected foreshortening and image resampling can be used.*

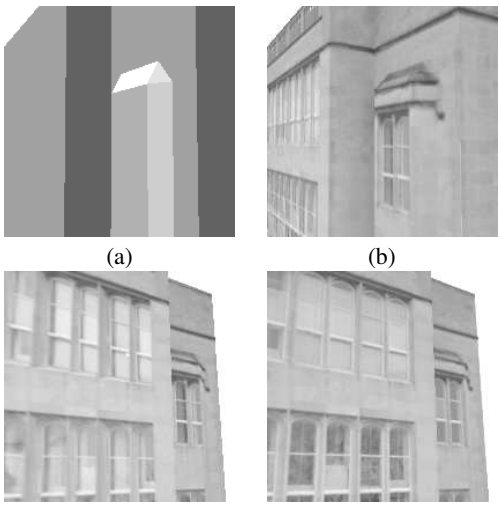


Figure 13: *View-dependent texture mapping.* **(a)** *A detail view of the high school model.* **(b)** *A rendering of the model from the same position using view-dependent texture mapping. Note that although the model does not capture the slightly recessed windows, the windows appear properly recessed because the texture map is sampled primarily from a photograph which viewed the windows from approximately the same direction.* **(c)** *The same piece of the model viewed from a different angle, using the same texture map as in (b). Since the texture is not selected from an image that viewed the model from approximately the same angle, the recessed windows appear unnatural.* **(d)** *A more natural result obtained by using view-dependent texture mapping. Since the angle of view in (d) is different than in (b), a different composition of original images is used to texture-map the model.*

address this, the pixel weights are ramped down near the boundary of the projected images. Although this method does not guarantee smooth transitions in all cases, we have found that it eliminates most artifacts in renderings and animations arising from such seams.

If an original photograph features an unwanted car, tourist, or other object in front of the architecture of interest, the unwanted object will be projected onto the surface of the model. To prevent this from happening, the user may mask out the object by painting over the obstruction with a reserved color. The rendering algorithm will then set the weights for any pixels corresponding to the masked regions to zero, and decrease the weights of the pixels near the boundary as before to minimize seams. Any regions in the composite image which are occluded in every projected image are filled in using the hole-filling method from [23].

In the discussion so far, projected image weights are computed at every pixel of every projected rendering. Since the weighting function is smooth (though not constant) across flat surfaces, it is not generally not necessary to compute it for every pixel of every face of the model. For example, using a single weight for each face of the model, computed at the face's center, produces acceptable results. By coarsely subdividing large faces, the results are visually indistinguishable from the case where a unique weight is computed for every pixel. Importantly, this technique suggests a real-time implementation of view-dependent texture mapping using a texture-mapping graphics pipeline to render the projected views, and α-channel blending to composite them.

For complex models where most images are entirely occluded for the typical view, it can be very inefficient to project every original photograph to the novel viewpoint. Some efficient techniques to determine such visibility *a priori* in architectural scenes through spatial partitioning are presented in [18].

4 Model-Based Stereopsis

The modeling system described in Section 2 allows the user to create a basic model of a scene, but in general the scene will have additional geometric detail (such as friezes and cornices) not captured in the model. In this section we present a new method of recovering such additional geometric detail automatically through stereo correspondence, which we call *model-based* stereo. Model-based stereo differs from traditional stereo in that it measures how the actual scene deviates from the approximate model, rather than trying to measure the structure of the scene without any prior information. The model serves to place the images into a common frame of reference that makes the stereo correspondence possible even for images taken from relatively far apart. The stereo correspondence information can then be used to render novel views of the scene using image-based rendering techniques.

As in traditional stereo, given two images (which we call the *key* and *offset*), model-based stereo computes the associated depth map for the key image by determining corresponding points in the key and offset images. Like many stereo algorithms, our method is *correlation-based*, in that it attempts to determine the corresponding point in the offset image by comparing small pixel neighborhoods around the points. As such, correlation-based stereo algorithms generally require the neighborhood of each point in the key image to resemble the neighborhood of its corresponding point in the offset image.

The problem we face is that when the key and offset images are taken from relatively far apart, as is the case for our modeling method, corresponding pixel neighborhoods can be foreshortened very differently. In Figs. 14(a) and (c), pixel neighborhoods toward the right of the key image are foreshortened horizontally by nearly a factor of four in the offset image.

The key observation in model-based stereo is that even though two images of the same scene may appear very different, they appear similar after being projected onto an approximate model of the scene. In particular, projecting the offset image onto the model and viewing it from the position of the key image produces what we call the *warped offset* image, which appears very similar to the key image. The geometrically detailed scene in Fig. 14 was modeled as two flat surfaces with our modeling program, which also determined the relative camera positions. As expected, the warped offset image (Fig. 14(b)) exhibits the same pattern of foreshortening as the key image.

In model-based stereo, pixel neighborhoods are compared between the key and warped offset images rather than the key and off-

(a) *Key Image* (b) *Warped Offset Image* (c) *Offset Image* (d) *Computed Disparity Map*

Figure 14: **(a)** *and* **(c)** *Two images of the entrance to Peterhouse chapel in Cambridge, UK. The Façade program was used to model the façade and ground as a flat surfaces and to recover the relative camera positions.* **(b)** *The warped offset image, produced by projecting the offset image onto the approximate model and viewing it from the position of the key camera. This projection eliminates most of the disparity and foreshortening with respect to the key image, greatly simplifying stereo correspondence.* **(d)** *An unedited disparity map produced by our model-based stereo algorithm.*

set images. When a correspondence is found, it is simple to convert its disparity to the corresponding disparity between the key and offset images, from which the point's depth is easily calculated. Fig. 14(d) shows a disparity map computed for the key image in (a).

The reduction of differences in foreshortening is just one of several ways that the warped offset image simplifies stereo correspondence. Some other desirable properties of the warped offset image are:

- Any point in the scene which lies on the approximate model will have zero disparity between the key image and the warped offset image.
- Disparities between the key and warped offset images are easily converted to a depth map for the key image.
- Depth estimates are far less sensitive to noise in image measurements since images taken from relatively far apart can be compared.
- Places where the model occludes itself relative to the key image can be detected and indicated in the warped offset image.
- A linear epipolar geometry (Sec. 4.1) exists between the key and warped offset images, despite the warping. In fact, the epipolar lines of the warped offset image coincide with the epipolar lines of the key image.

4.1 Model-Based Epipolar Geometry

In traditional stereo, the *epipolar constraint* (see [6]) is often used to constrain the search for corresponding points in the offset image to searching along an epipolar line. This constraint simplifies stereo not only by reducing the search for each correspondence to one dimension, but also by reducing the chance of selecting a false matches. In this section we show that taking advantage of the epipolar constraint is no more difficult in model-based stereo case, despite the fact that the offset image is non-uniformly warped.

Fig. 15 shows the epipolar geometry for model-based stereo. If we consider a point P in the scene, there is a unique *epipolar plane* which passes through P and the centers of the key and offset cameras. This epipolar plane intersects the key and offset image planes in *epipolar lines* e_k and e_o. If we consider the projection p_k of P onto the key image plane, the epipolar constraint states that the corresponding point in the offset image must lie somewhere along the offset image's epipolar line.

In model-based stereo, neighborhoods in the key image are compared to the warped offset image rather than the offset image. Thus, to make use of the epipolar constraint, it is necessary to determine where the pixels on the offset image's epipolar line project to in the warped offset image. The warped offset image is formed by projecting the offset image onto the model, and then reprojecting the model onto the image plane of the key camera. Thus, the projection p_o of P in the offset image projects onto the model at Q, and then reprojects to q_k in the warped offset image. Since each of these projections occurs within the epipolar plane, any possible correspondence

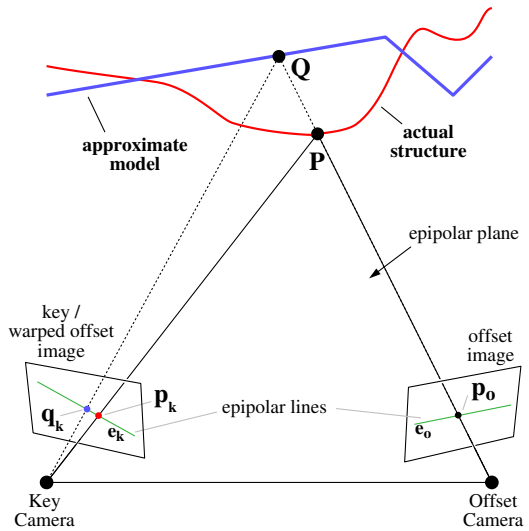


Figure 15: *Epipolar geometry for model-based stereo.*

for p_k in the key image must lie on the *key* image's epipolar line in the warped offset image. In the case where the actual structure and the model coincide at P, p_o is projected to P and then reprojected to p_k, yielding a correspondence with zero disparity.

The fact that the epipolar geometry remains linear after the warping step also facilitates the use of the ordering constraint [2, 6] through a dynamic programming technique.

4.2 Stereo Results and Rerendering

While the warping step makes it dramatically easier to determine stereo correspondences, a stereo algorithm is still necessary to actually determine them. The algorithm we developed to produce the images in this paper is described in [3].

Once a depth map has been computed for a particular image, we can rerender the scene from novel viewpoints using the methods described in [23, 16, 13]. Furthermore, when several images and their corresponding depth maps are available, we can use the view-dependent texture-mapping method of Section 3 to composite the multiple renderings. The novel views of the chapel façade in Fig. 16 were produced through such compositing of four images.

5 Conclusion and Future Work

To conclude, we have presented a new, photograph-based approach to modeling and rendering architectural scenes. Our modeling approach, which combines both geometry-based and image-based modeling techniques, is built from two components that we have developed. The first component is an easy-to-use photogrammet-

Figure 16: *Novel views of the scene generated from four original photographs. These are frames from an animated movie in which the façade rotates continuously. The depth is computed from model-based stereo and the frames are made by compositing image-based renderings with view-dependent texture-mapping.*

ric modeling system which facilitates the recovery of a basic geometric model of the photographed scene. The second component is a model-based stereo algorithm, which recovers precisely how the real scene differs from the basic model. For rendering, we have presented view-dependent texture-mapping, which produces images by warping and compositing multiple views of the scene. Through judicious use of images, models, and human assistance, our approach is more convenient, more accurate, and more photorealistic than current geometry-based or image-based approaches for modeling and rendering real-world architectural scenes.

There are several improvements and extensions that can be made to our approach. First, surfaces of revolution represent an important component of architecture (e.g. domes, columns, and minarets) that are not recovered in our photogrammetric modeling approach. (As noted, the dome in Fig. 10 was manually sized by the user.) Fortunately, there has been much work (e.g. [24]) that presents methods of recovering such structures from image contours. Curved model geometry is also entirely consistent with our approach to recovering additional detail with model-based stereo.

Second, our techniques should be extended to recognize and model the photometric properties of the materials in the scene. The system should be able to make better use of photographs taken in varying lighting conditions, and it should be able to render images of the scene as it would appear at any time of day, in any weather, and with any configuration of artificial light. Already, the recovered model can be used to predict shadowing in the scene with respect to an arbitrary light source. However, a full treatment of the problem will require estimating the photometric properties (i.e. the bidirectional reflectance distribution functions) of the surfaces in the scene.

Third, it is clear that further investigation should be made into the problem of selecting which original images to use when rendering a novel view of the scene. This problem is especially difficult when the available images are taken at arbitrary locations. Our current solution to this problem, the weighting function presented in Section 3, still allows seams to appear in renderings and does not consider issues arising from image resampling. Another form of view selection is required to choose which pairs of images should be matched to recover depth in the model-based stereo algorithm.

Lastly, it will clearly be an attractive application to integrate the models created with the techniques presented in this paper into forthcoming real-time image-based rendering systems.

Acknowledgments

This research was supported by a National Science Foundation Graduate Research Fellowship and grants from Interval Research Corporation, the California MICRO program, and JSEP contract F49620-93-C-0014. The authors also wish to thank Tim Hawkins, Carlo Séquin, David Forsyth, and Jianbo Shi for their valuable help in revising this paper.

References

[1] Ali Azarbayejani and Alex Pentland. Recursive estimation of motion, structure, and focal length. *IEEE Trans. Pattern Anal. Machine Intell.*, 17(6):562–575, June 1995.

[2] H. H. Baker and T. O. Binford. Depth from edge and intensity based stereo. In *Proceedings of the Seventh IJCAI, Vancouver, BC*, pages 631–636, 1981.

[3] Paul E. Debevec, Camillo J. Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. Technical Report UCB//CSD-96-893, U.C. Berkeley, CS Division, January 1996.

[4] D.J.Fleet, A.D.Jepson, and M.R.M. Jenkin. Phase-based disparity measurement. *CVGIP: Image Understanding*, 53(2):198–210, 1991.

[5] Oliver Faugeras and Giorgio Toscani. The calibration problem for stereo. In *Proceedings IEEE CVPR 86*, pages 15–20, 1986.

[6] Olivier Faugeras. *Three-Dimensional Computer Vision*. MIT Press, 1993.

[7] Olivier Faugeras, Stephane Laveau, Luc Robert, Gabriella Csurka, and Cyril Zeller. 3-d reconstruction of urban scenes from sequences of images. Technical Report 2572, INRIA, June 1995.

[8] W. E. L. Grimson. *From Images to Surface*. MIT Press, 1981.

[9] D. Jones and J. Malik. Computational framework for determining stereo correspondence from a set of linear spatial filters. *Image and Vision Computing*, 10(10):699–708, December 1992.

[10] E. Kruppa. Zur ermittlung eines objectes aus zwei perspektiven mit innerer orientierung. *Sitz.-Ber. Akad. Wiss., Wien, Math. Naturw. Kl., Abt. Ila.*, 122:1939–1948, 1913.

[11] H.C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293:133–135, September 1981.

[12] D. Marr and T. Poggio. A computational theory of human stereo vision. *Proceedings of the Royal Society of London*, 204:301–328, 1979.

[13] Leonard McMillan and Gary Bishop. Plenoptic modeling: An image-based rendering system. In *SIGGRAPH '95*, 1995.

[14] Eric N. Mortensen and William A. Barrett. Intelligent scissors for image composition. In *SIGGRAPH '95*, 1995.

[15] S. B. Pollard, J. E. W. Mayhew, and J. P. Frisby. A stereo correspondence algorithm using a disparity gradient limit. *Perception*, 14:449–470, 1985.

[16] R. Szeliski. Image mosaicing for tele-reality applications. In *IEEE Computer Graphics and Applications*, 1996.

[17] Camillo J. Taylor and David J. Kriegman. Structure and motion from line segments in multiple images. *IEEE Trans. Pattern Anal. Machine Intell.*, 17(11), November 1995.

[18] S. J. Teller, Celeste Fowler, Thomas Funkhouser, and Pat Hanrahan. Partitioning and ordering large radiosity computations. In *SIGGRAPH '94*, pages 443–450, 1994.

[19] Carlo Tomasi and Takeo Kanade. Shape and motion from image streams under orthography: a factorization method. *International Journal of Computer Vision*, 9(2):137–154, November 1992.

[20] Roger Tsai. A versatile camera calibration technique for high accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. *IEEE Journal of Robotics and Automation*, 3(4):323–344, August 1987.

[21] S. Ullman. *The Interpretation of Visual Motion*. The MIT Press, Cambridge, MA, 1979.

[22] L Williams. Casting curved shadows on curved surfaces. In *SIGGRAPH '78*, pages 270–274, 1978.

[23] Lance Williams and Eric Chen. View interpolation for image synthesis. In *SIGGRAPH '93*, 1993.

[24] Mourad Zerroug and Ramakant Nevatia. Segmentation and recovery of shgcs from a real intensity image. In *European Conference on Computer Vision*, pages 319–330, 1994.