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

Technical Report UCB//CSD-96-893
January 19, 1996

Paul E. Debevec	Camillo J. Taylor	Jitendra Malik
debevec@cs.berkeley.edu	camillo@cs.berkeley.edu	malik@cs.berkeley.edu
545 Soda Hall	485 Soda Hall	725 Soda Hall
(510) 642 9940	(510) 642 5029	(510) 642 7597

Computer Science Division, University of California at Berkeley
Berkeley, CA 94720-1776
(510) 642 5775 (fax)

Abstract

We present an approach for creating realistic synthetic views of existing architectural scenes from a sparse set of still photographs. Our approach, which combines both geometry-based and image-based modeling and rendering techniques, has two components. The first component is an easy-to-use photogrammetric modeling system which facilitates the recovery of a basic geometric model of the photographed scene. The modeling system is effective 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 approach can robustly recover accurate depth from image pairs with large baselines. Consequently, our approach can model large architectural environments with far fewer photographs than current image-based modeling approaches. As an intermediate result, we present *view-dependent texture mapping*, a method of better simulating geometric detail on basic models. Our approach can recover models for use in either geometry-based or image-based rendering systems. We present results that demonstrate our approach's abilty to create realistic renderings of architectural scenes from viewpoints far from the original photographs.

Keywords: Image-based modeling, image-based rendering, interactive modeling systems, photogrammetry, reconstruction, view-dependent texture mapping, view interpolation, model-based stereo

See also: http://www.cs.berkeley.edu/~debevec/Research/

1 Introduction

As the technology for experiencing immersive virtual environments develops, there will be a growing need for interesting virtual environments to experience. So far, computer models of architectural scenes have been especially popular subjects, mirroring the real-world popularity of architectural sites as places to visit. From this, there is a call for a method to conveniently acquire photorealistic models of existing architecture.

The creation of three-dimensional models of existing architectural scenes with the aid of the computer has been commonplace for some time, and the resulting models have proven to be entertaining virtual environments as well as valuable visualization tools. Large-scale efforts have pushed the campuses of Iowa State University, California State University – Chico, and swaths of downtown Los Angeles [15] through the graphics pipeline. Unfortunately, the modeling methods employed in such projects are very labor-intensive. They typically involve surveying the site, locating and digitizing architectural plans (if available), and converting existing CAD data (again, if available). Moreover, the renderings of such models are noticeably computer-generated; even those that employ liberal texture-mapping generally fail to resemble real 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 has an advantage in producing photorealistic renderings as output. Some of the most promising of these systems [31, 22] 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 which recover the structure of the scene. This has caused problems because state-of-the-art stereo algorithms have a number of significant weaknesses; in particular, the images analyzed by the algorithm need to be very similar for useful results to be obtained. Because of this, the image-based techniques presented have used images that were taken a short distance apart relative to the depth of objects in the scene, and in the case of [22], have also employed significant amounts of user input (see section 2.4) per image pair. These concessions to the weakness of stereo algorithms bode poorly for creating large-scale, freely navigable virtual environments from photographs. First, capturing the data for a realistically renderable model could require an impractical number of closely spaced photographs, and second, deriving the depth from the photographs could require an impractical amount of user input.

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 an approach that combines the strengths of both geometry-based and image-based methods, as illustrated in Fig. 1. 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, and the remaining geometric detail is recovered automatically through stereo correspondence. The final images can be rendered with current image-based rendering techniques. Because only photographs are required, our approach is neither invasive nor does it require architectural plans, CAD models, or specialized instrumentation such as surveying equipment, GPS sensors or range

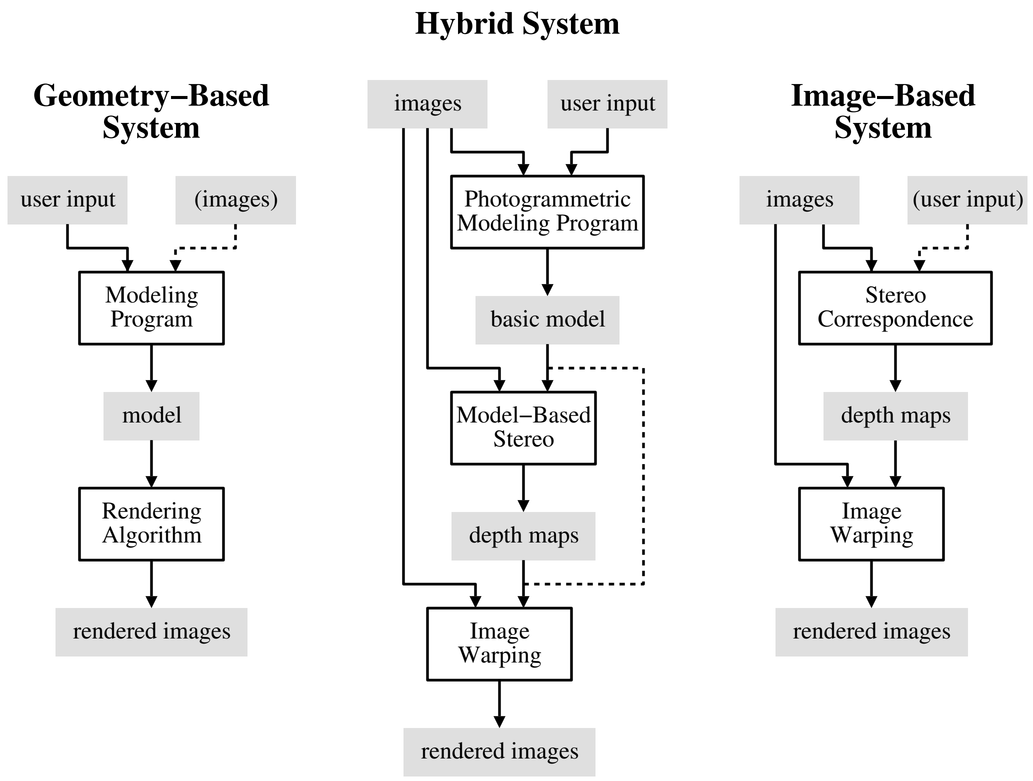


Figure 1: Schematic of how our hybrid approach (center) combines geometry-based and image-based approaches to modeling and rendering architecture from photographs. The geometry-based approach illustrated places the majority of the modeling task on the user, whereas the image-based approach places the majority of the task on the computer. Our method divides the modeling task into two stages, one that is interactive, and one that is automated. The dividing point we have chosen capitalizes on the strengths of both the user and the computer to produce the best possible models and renderings using the fewest number of photographs.
The dashed line in the geometry-based schematic indicates that images may optionally be used in a modeling program as texture-maps. The dashed line in the image-based schematic indicates that in some systems user input is used to initialize the stereo correspondence algorithm. The dashed line in the hybrid schematic indicates that view-dependent texture-mapping (as discussed in Section 5) can be used without performing stereo correspondence.

scanners.

2 Background and Related Work

The process of reconstructing three-dimensional structures from two-dimensional images has long been a principal endeavour of the field of computer vision, and the process of rendering such recovered structures is a subject which has recently received 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: Determining Structure from Multiple Views, Stereo Correspondence, Modeling from Range Images, and Image-Based Rendering.

2.1 Determining Structure from Multiple Views

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, the discipline of measuring the world using photographic images, mainly for producing topographic maps. Back in 1913, Kruppa [18] 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). More recently, the problem has been studied vigorously in computer vision under the title of *structure from motion*. Both its mathematical and algorithmic aspects have been explored starting from the fundamental work of Ullman [38] and Longuet-Higgins [20], in the early 1980s. Faugeras's book [10] provides a synthesis of the state of the art as of 1992. A key realization was that the recovery of structure is quite sensitive to noise in image measurements, and that the problem is made additionally difficult when the translation of the camera is small or when the scene points lie in nearly degenerate configurations.

Attention has turned to using more than two views with image stream methods such as [35] or recursive approaches [39, 6]. When a scaled orthography approximation is appropriate, [35] has shown excellent results, but a general solution for the case of perspective projection remains 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 tailored to structured environments [33] has used line segments instead of points for the as image features, 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 and individual line segments, which are not directly renderable as photorealistic 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 overcome the problems of numerical instability that plague such approaches. We demonstrate our approach's successful use in an interactive system for building architectural models from photographs.

2.1.1 Camera Calibration

Determining structure from multiple views is a simpler problem when the cameras are *calibrated*, that is, their *intrinsic* parameters (see [10]) are known. A camera's intrinsic parameters do not depend on its position in space and include the focal length, the center of projection, and the aspect ratio of the image pixels. Real camera lenses can also introduce significant nonlinear radial distortion into images (an effect modeled in [17]), which is usually well-approximated by a simple parametric mapping. A camera's *extrinsic* parameters, in contrast, relate the camera's frame of reference to the world coordinate system by a 3D rigid transformation. Camera calibration is a well-studied problem both in photogrammetry and computer vision; successful methods include [36] and [9]. Recent results in computer vision [8] recover projective structure without calibrated cameras, but can not recover all aspects of affine and euclidean structure without calibration information. In our work we have used calibrated cameras.

2.2 Computational Stereo: determining correspondences in multiple views

The geometrical theory of structure form multiple views relies on being able to solve the *correspondence* problem, which is to identify the points in two images that are projections of the same point in the world. Humans solve this problem, seemingly effortlessly, in the process of binocular stereopsis. Two slightly different pictures of the world are imaged on the two retinas and the visual cortex solves the correspondence problem to produce a vivid perception of depth. Two terms used in reference to stereo are *baseline* and *disparity*. The baseline of a stereo pair is the distance between the camera locations of the two images. Disparity refers to the difference in image location between corresponding features in the two images, which is projectively related to the depth of the feature in the scene.

Solving the correspondence problem with computer algorithms has proved hard in spite of years of research on the problem [3, 7, 11, 16, 21, 25, 26]. The major sources of difficulty include:

1. **Foreshortening**. Surfaces in the scene viewed from different positions will be foreshortened differently in the images, causing the image neighborhoods of corresponding pixels to appear dissimilar. Such dissimilarity can confound stereo algorithms that use local similarity metrics to determine correspondences.

2. **Occlusions**. Depth discontinuities in the world can create half-occluded regions in an image pair, which also poses problems for local similarity metrics.

3. **Lack of Texture**. Where there is an absence of image intensity features it is difficult for a stereo algorithm to correctly find the correct match for a particular point, since

many point neighborhoods will be similar in appearance.

In our approach, we address all three of these problems by making use of the geometric model recovered in our interactive modeling system. The model enables us to warp the images to eliminate unequal foreshortening and to predict major instances of occlusion *before* trying to find correspondences; it also suggests a reasonable default disparity value when there is a lack of texture in the scene. Occlusions are further handled during the stereo correspondence process by an iterative grouping and disparity refinement stage. In section 6 we show how the algorithm is able to successfully determine stereo correspondences, even with baselines that are large with respect to the distance of the objects in the scene.

2.3 Modeling from Range Images

Instead of the anthropomorphic approach of using multiple images to reconstruct scene structure, an alternative technique is to use range imaging sensors [4] to directly measure depth to various points in the scene. Early versions of these sensors were slow, cumbersome and expensive. Altough many improvements have been made, the most convenient application currently is for human scale objects and not for architectural scenes. Algorithms for combining multiple range images from different viewpoints have been developed both in computer vision [42, 30, 28] and in computer graphics [14, 37]. In many ways, range image based techniques and photographic techniques are complementary and have their relative advantages and disadvantages. Some advantages of modeling from photographic images are that (a) 35mm cameras are inexpensive and widely available and (b) for some architecture that no longer exists all that is available are photographs. Furthermore, range images alone are insufficient for producing renderings of a scene; photometric information from photographs is also necessary.

2.4 Image-Based Rendering

In an image-based rendering system, the model consists of a set of images of a scene and their corresponding z-buffers, or *depth maps*. The key observation in image-based rendering is that 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 into a new virtual camera. Thus, the computer generates a new image of the scene by warping the data in the images according to their depth values. 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, [41] demonstrated how regularly spaced synthetic images (with their computed depth maps) could be warped and composited in real time to produce a virtual environment.

Recently, [22] presented a real-time image-based rendering system that used panoramic photographs with depth computed, in part, from stereo correspondence. The system successfully demonstrated how a single real image could be re-rendered from nearby points of view. A principal 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 taken from close together: 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 (approximately 300 by 300 feet) would require 3,600 panoramic images spaced five feet apart. While image compression could make the storage of such a large amount of image data possible, acquiring so many photographs is clearly 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 photographs.

3 Overview

In this section we outline the main components of our approach to modeling and rendering architectural scenes from photographs. The key insight that distinguishes our approach from a standard computer vision structure from motion/stereopsis formulation (and enables us to bypass the difficulties discussed in Section 2) is that we are dealing with a particular domain—architectural scenes—which has a particular set of characteristics that can be used to make the problem simpler. Instead of posing the problem as one of recovering the 3D positions of an arbitrary set of features, we formulate the problem as recovering a model of the scene at two levels of resolution:

1. **A coarse model of the scene**. The coarse model is recovered as an arrangement of geometric primitives via an interactive photogrammetric modeling system. For this model, geometric details such as friezes, molding, and statuary are ignored. The result is a model with relatively few parameters (e.g. forty-five for the clock tower model in Fig. 2) that one can recover robustly from the images of the scene. Since the scene model contains relatively few parameters, the reconstruction is very robust, unlike a traditional structure-from-multiple-views algorithm.

2. **A detailed model of the scene**. Here the geometric details that were ignored in the interactive modeling stage are computed as displacements from the coarse model. To recover these displacements, we use a *model-based* stereo algorithm. Unlike previous approaches, we exploit the availability of a coarse model to simplify the problem of computing stereo correspondences: the model is used to *pre-warp* the images to factor out unequal foreshortening and and eliminates all image disparity where there are no deviations from the coarse model. As a principal result, we show that this warping operation is as simple as projecting the images onto the model.

Our approach is successful not only because it synthesizes these geometry-based and image-base techniques, but because it splits the task of modeling from images into sub-tasks which are easily accomplished by a person (but not a computer algorithm), and sub-tasks which are easily performed by a computer algorithm (but not a person.) The correspondences for the reconstruction of the coarse model of the system are provided by the user in a interactive way; for this purpose we have designed and implemented *Façade* (Section 4), a photogrammetric modeling system that makes this task quick and simple. Typically, the correspondences the user must provide are few in number per image and easy to observe. By design, the high-level model recoved by the system is precisely the sort of scene information that would be difficult for a computer algorithm to discover automatically. The geometric detail recovery is performed by an automated stereo correspondence algorithm (Section 6), which has been made feasible and robust by the pre-warping step provided by the coarse geometric model. In this case, corresponding points must be computed for a dense sampling of image pixels, a job too tedious to assign to a human, but feasible for a computer to perform using model-based stereo.

Of course, many computer graphics systems (e.g. [1]) and applications (e.g. *Doom* by Id Software) already make use of a well-known technique to simulate detail in coarse geometric models: texture-mapping. As an intermediate result (Section 5), we relate our work to the process of texture mapping, and develop a new rendering method, called *view-dependent texture mapping*, that uses textures obtained from multiple views of the model to provide a better illusion of depth than traditional texture mapping allows.

Our synthesis of geometry- and image-based techniques can create realistic renderings of an architectural scene with a relatively sparse set of photographs (for example, three photographs are used for the rendering in Fig. 3, twelve for Fig. 12, and four for Fig. 19). To be more specific, we present a method to accurately recover dense depth measurements using images that are taken far apart relative to the objects in the scene. Such large baselines allow high-precision scene depth measurement, which makes it possible to render coherent novel views far away from the original camera locations. In the absence of this framework, the only demonstrated way to create freely navigable environments from photographs is to acquire an inconveniently large number of closely spaced photographs.

4 The Interactive Modeling System

In this section we present *Façade*, a simple interactive modeling system that allows a user to construct a geometric model of a scene from a set of digitized photographs. In Façade, the user constructs a parameterized geometric model of the scene while the program computes the parameters that best make the model conform to the photographs. We first describe Façade's user interface, and then present the underlying model representation and algorithms.

4.1 User Interface

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 shows the two types of windows used in the Façade program: image viewers and model viewers. The user supplies input to the program by instancing the components of the model, marking features of interest in the images, and indicating which features in the images correspond to which features in the model. Façade then computes the sizes and relative positions of the components of the model that best fit the features marked in the photographs.

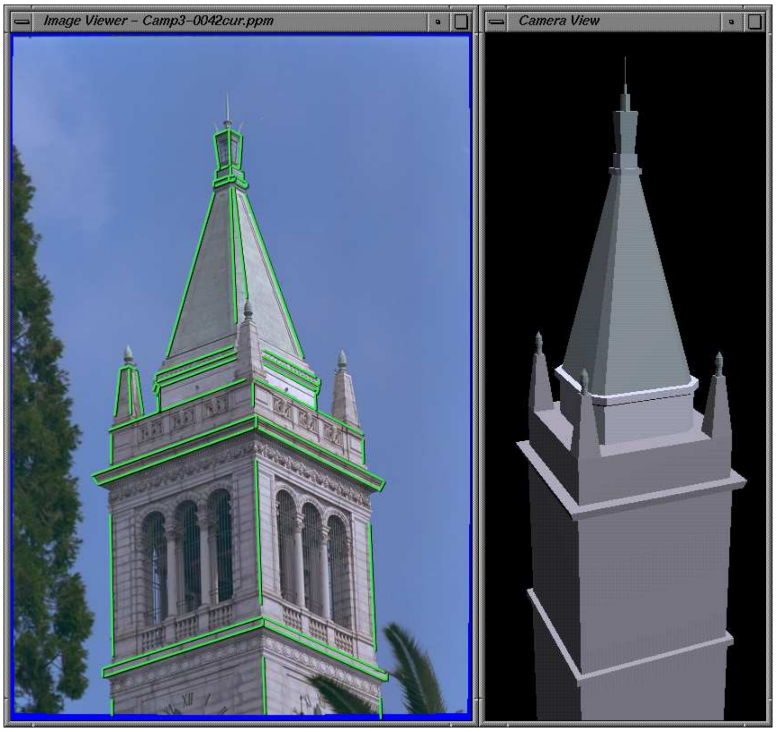


Figure 2: An image viewer and a model viewer from the Façade modeling system. The left window shows a single photograph of Berkeley's clock tower, the Campanile, with features the user has marked shown in green. The right window shows the recovered model, the parameters of which have been computed by Façade's reconstruction algorithm. Note that only the left pinnacle has been marked—the remaining three (including one not visible) have been recovered by encoding symmetry into the model. Although Façade allows any number of images to be used, in this case the constraints in the model have made it possible to recover an accurate 3D model from a single photograph.

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 each part of the scene as such a primitive; the user may also create new block classes if desired. What the user does not need to specify are the numerical values of the blocks' parameters; these parameters are recovered automatically from the features marked in the photographs.

The user may choose to constrain the sizes and positions of any of the blocks. In Fig. 2, most of the blocks have been constrained to have equal length and width. Additionally, the four pinnacles have been constrained to have the same dimensions. Blocks may also be placed in constrained relations to one other. For example, many of the blocks in Fig. 2 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 given, they are automatically used by Façade to simplify the reconstruction problem.

The user marks edge features in the images using a simple point-and-click interface; features may be marked with sub-pixel accuracy by zooming into the images. Façade uses edge rather than point features since they are easier to localize and less likely to be completely obscurred. Only a section of any particular edge needs to be marked, so it is possible to make use of partially visible edges. Façade is able to compute accurate reconstructions with only a portion of the visible edges marked in any particular image, particularly when the user has embedded constraints in the model. We have also investigated a gradient-based technique as in [23] to further assist the user.

With the edges marked, the user needs to specify which features in the model correspond to which features in the images. This is accomplished by clicking on an edge in the model and then clicking on the corresponding edge in one of the images. The user can rotate their view of the model into a perferred orientation to make the process of forming correspondences easier.

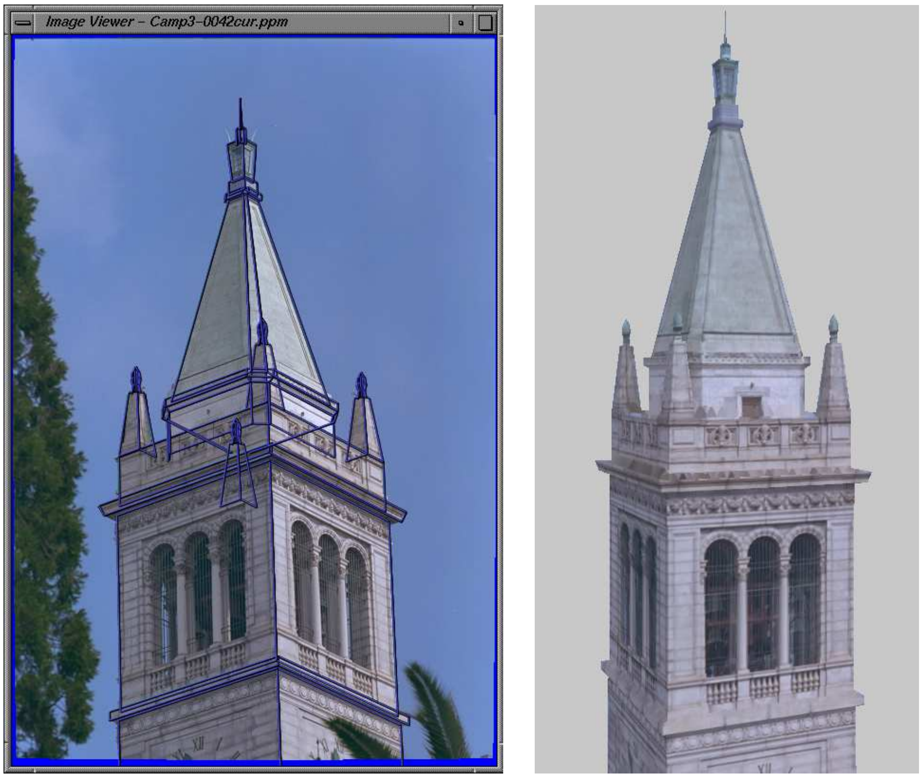
At any time, the user may instruct the computer to reconstruct the model. The computer then solves for the parameters of the model that cause it to align with the observed features in the images. During the reconstruction, the computer also determines and displays the locations from which the original photographs were taken. For simple models consisting of just a few blocks, a full reconstruction takes only a few seconds; for more complex models, a full reconstruction can take a few minutes. For this reason, the user can instruct the computer to employ faster but less precise reconstruction algorithms 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. 3(a) shows the results of projecting the edges of the model in Fig. 2 into the original image.

Lastly, the user may generate novel views of the model by positioning a virtual camera at any desired location. Façade will then use either the view-dependent texture-mapping method of Section 5 or the detail recovery method of Section 6 to render a novel view of the scene from the desired location. Fig. 2(b) shows an aerial rendering of the model using the view-dependent texture-mapping method.

4.2 Model Representation

In Façade, an architectural scene is represented as a set of polyhedral blocks. Each block has a set of parameters which define its size and shape. The coordinates of the vertices of these polyhedra with respect to the block's internal frame of reference are expressed as a linear functions of the block parameters. For example, in the wedge primitive shown in Fig. 4, the coordinates of the vertex P_o can be computed using the expression $P_o = (-width, -height, length)^T$. Each block also has an associated bounding box whose exents are also expressed as linear functions of the block parameters. For example, the minimum x-extent of the block shown in Fig. 4, $wedge_x^{MIN}$, can be computed from the expression $wedge_x^{MIN} = -width$.



(a) (b)

Figure 3: (a) The results of reprojecting the recovered model from Fig. 2 onto the original photograph through the recovered camera position. The model conforms precisely to the original image, which is an indication that the building has been accurately reconstructed. (b) A synthetic view of the Campanile generated using the view-dependent texture-mapping method described in Section 5. A real photograph from this position would be difficult to take since the camera position is 250 feet above the ground.

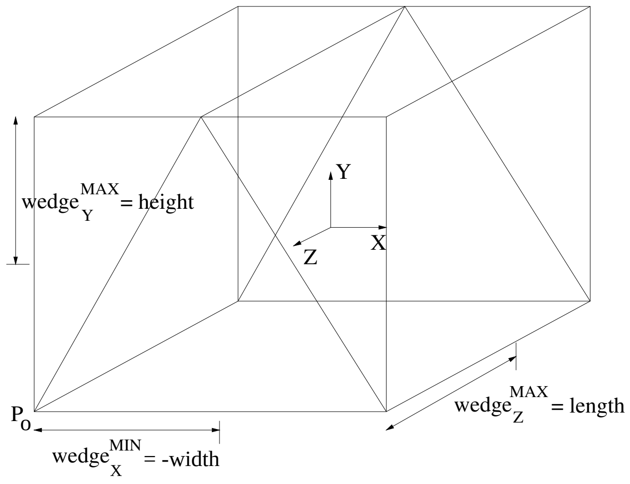


Figure 4: A wedge block with its associated internal parameters and bounding box.

Each class of block used in our modeling program is defined in a simple text file. Each block file specifies the parameters of the block, the parameter coefficients needed to compute the vertices of the block and the extents of its bounding box, and the connectivity of the block's vertices, edges, and faces. If desired, the user can add a custom block to their project by authoring (or procedurally generating) a new block file.

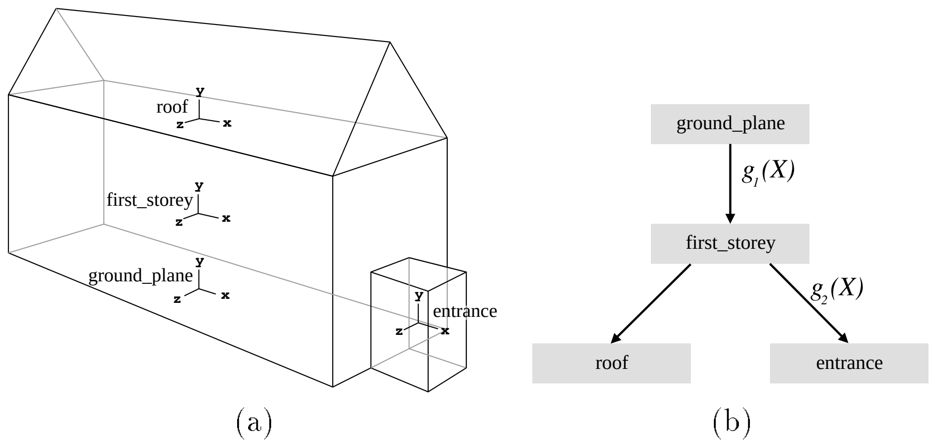


Figure 5: (a) A simple geometric model of a building. (b) The model's tree representation that would be used in Façade. The nodes in the tree represent geometric primitives while the links contain the spatial relationships between the blocks.

The blocks in Façade are organized in a hierarchical tree structure as shown in Fig. 5(b). Each node of the tree represents an individual block, except for the root of the tree which serves only to establish the world coordinate frame. The links in the tree contain the spatial relationships between blocks, called *relations*. Similar tree representations are used throughout computer graphics to model complicated objects in terms of simple geometrical primitives.

The relation between a block and its parent can always be represented in terms of a rotation matrix R and a translation vector t. This representation requires six degrees of freedom: three for each of R and t. In architectural scenes, however, the relationship between two blocks often has a simple form that can be represented with fewer parameters, and Façade allows the user to embed 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:

1. An unconstrained rotation (three degrees of freedom)
2. A rotation about a particular coordinate axis (one degree of freedom)
3. No rotation, or a fixed rotation (zero degrees of freedom)

Façade also allows the user to specify constraints on each component of the translation vector t. The user can either leave the translation along a given dimension unconstrained, or force the bounding boxes of the two blocks to align themselves in some manner along that dimension. For example, in order to ensure that the roof block in Fig. 5 lies on top of the first storey block, the user could specify that the maximum y extent of the first storey block should be aligned with the minimum y extent of the roof block. With this constraint, the translation along the y axis is computed ($t_y = (first\_storey_y^{MAX} - roof_y^{MIN})$) rather than solved for.

Each parameter of each instantiated block is actually a reference to a named symbolic variable, as illustrated in Fig. 6. 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.

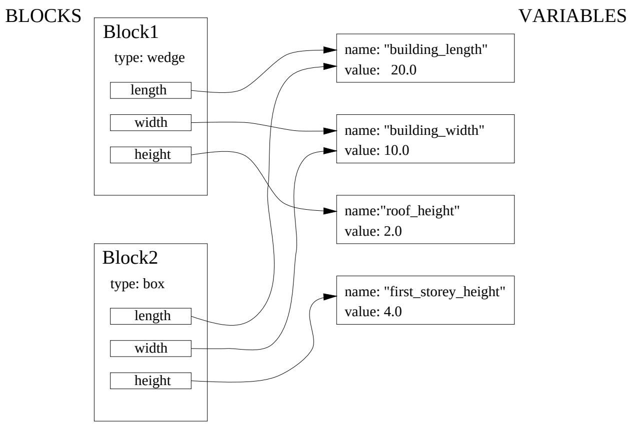


Figure 6: Implementation of the dimensions of a block primitive in terms of symbol references. A single variable can be referenced in multiple places in the model, allowing constraints of symmetry.

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. 5. 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 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 is illustrated best with an example. The model in Fig. 2 is parameterized by just 45 variables. 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 system as compared to traditional structure from motion algorithms.

4.3 Reconstruction Algorithm

The reconstruction algorithm used in Façade works by minimizing an objective function $\mathcal{O}$ that sums the disparity between the projected edges of the recovered model and the edges observed in the images, i.e. $\mathcal{O} = \sum Err_i$ where Err_i represents the disparity computed for edge feature i. Estimates for the unknown model parameters and camera positions are obtained by minimizing the objective function with respect to these variables. Our system uses the the error function Err_i presented in [33], which is described below.

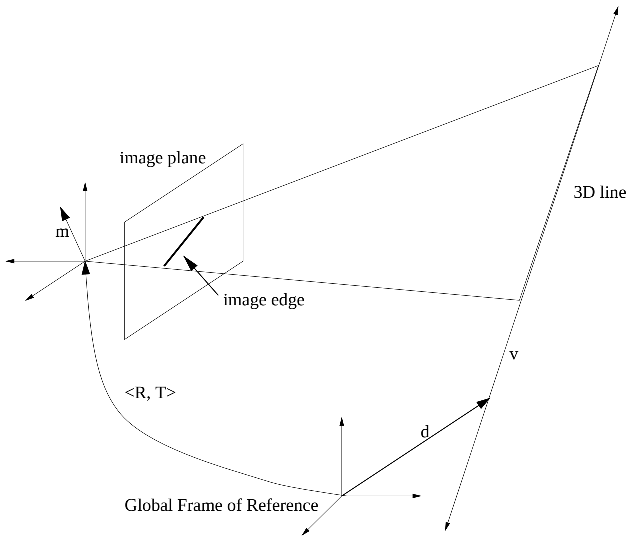


Figure 7: Projection of a straight line onto a camera's image plane.

Fig. 7 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 by 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. 8 shows how the disparity 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.

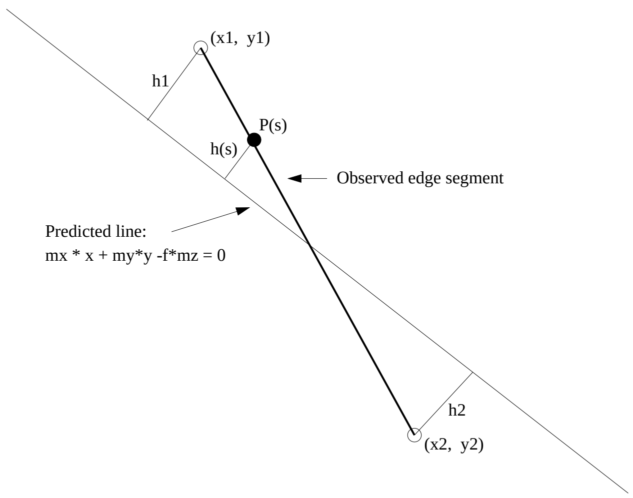


Figure 8: The error function used in the reconstruction algorithm. The heavy line represents the observed edge segment (marked by the user) and the lighter line represents the model edge predicted by the current camera and model parameters.

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

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

where:

$$\begin{aligned} \mathbf{m} &= (m_x, m_y, m_z)^T \\ A &= \begin{pmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \end{pmatrix} \end{aligned}$$

$$B = \frac{l}{3(m_x^2 + m_y^2)} \begin{pmatrix} 1 & 0.5 \\ 0.5 & 1 \end{pmatrix}$$

The objective function $\mathcal{O}$ is simply the sum of these error terms. This non-linear objective function is minimized using a variant of the Newton-Raphson method described in [32]. Our optimization technique is also similar to the one developed in [29].

The minimization procedure involves calculating the gradient and Hessian of the objective function 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 and evaluates them to obtain the gradient and Hessian at each step. This procedure is computationally inexpensive since the expressions for d and v obtained from Equations 2 and 1 are not complicated.

4.4 Computing an Initial Estimate

The objective function described in the previous 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 could just as easily converge to a local minimum instead of the globally best solution. In order 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 globally best solution. In practice, our initial estimate method consistently keeps the nonlinear minimization algorithm from converging to a local minimum.

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:

$$m^T R_j \mathbf{v} = 0 \tag{5}$$

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

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

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

From these equations, it can be deduced that any model edges of known orientation constrain the possible values for R_j. Since most architectural models contain many such edges, each camera rotation can be usually be directly estimated from the model independent of the model dimensions 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:

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

Once initial estimates for the camera rotations are computed, Equation 6 is used to obtain initial estimates for the remaining parameters determining the structure of the model and the locations of the cameras. 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.

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

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 initial estimates have been obtained, the non-linear minimization over the entire parameter space is applied to produce the best possible recovery of all the unknown parameters. Typically, the minimization requires fewer than ten iterations and adjusts the parameters of the model by at most a few percent. In our experience, the recovered models align closely with the original photographs. The next section presents such results.

4.5 Results

Figs. 2 and 3 showed the results of using Façade to reconstruct a clock tower from a single image. Figs. 9, 10, and 11 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 5.) 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. 10 shows that the recovered model conforms to the photographs to within a pixel, indicating an accurate reconstruction. Fig. 11 shows some views of the recovered model and camera positions.

5 Rendering the Model with 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 6 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.

Figure 9: Three of the twelve photographs used to reconstruct a high school building. The green lines indicate the edges the user has marked.

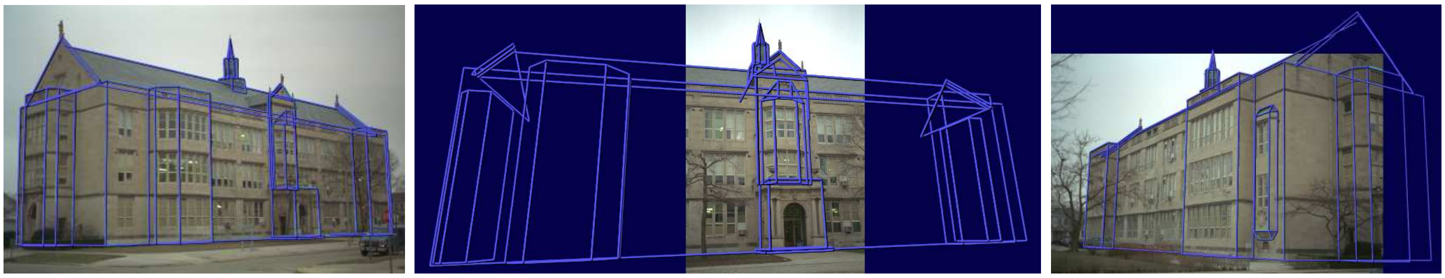
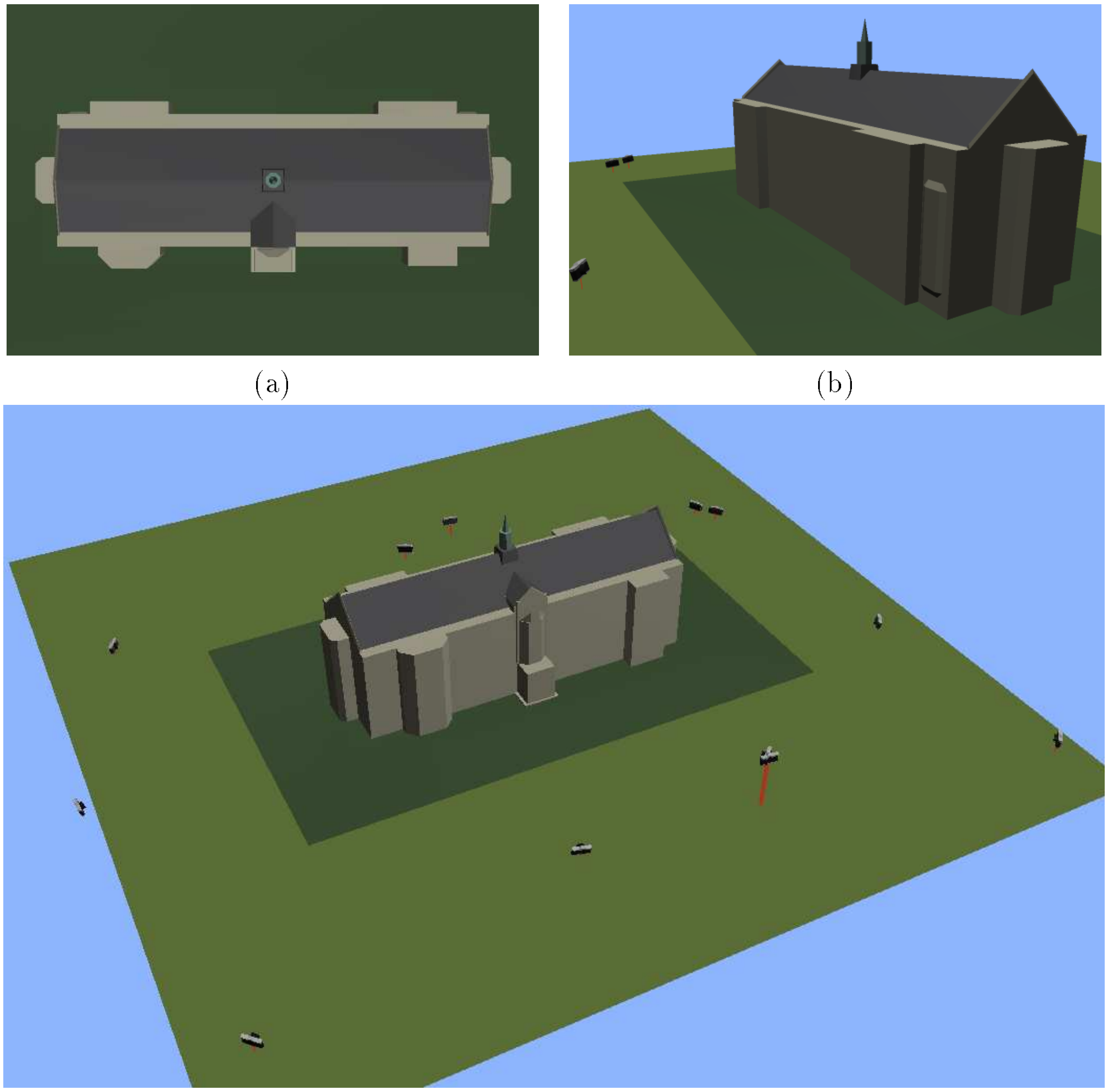


Figure 10: The edges of the reconstructed model, projected through the recovered camera positions and overlaid on the corresponding images. The recovered model conforms to the photographs to within a pixel in all twelve images, indicating that the building has been accurately reconstructed.



(a) (b)

(c)

Figure 11: A high school building model, reconstructed from twelve photographs. (a) Overhead view. (b) Rear view. (c) Aerial view showing the recovered original camera positions. Two nearly coincident cameras can be observed in front of the building; their corresponding photographs were taken from a second floor window of the building across the street.

Since the camera positions of the orginal 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.

5.1 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 [1]. When the model is not convex, it is possible that some parts of the model will shadow others with respect to the camera. Such shadowed regions could be determined using an object-space visible surface algorithm, such as [5], or an image-space ray casting algorithm, such as in [2]. We use an image-space shadow map algorithm based on [40] since such an algorithm can be implemented efficiently using standard polygon rendering hardware.

Fig. 12, 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. The original image has been resampled using bilinear interpolation; schemes less prone to aliasing are surveyed in [13].

5.2 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. 12 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 are not perfectly Lambertian[2]. In this

[1]In the art exhibit *Displacements* [24], Michael Naimark performed such a replacement literally. The model he used was the scene itself, painted white to catch the light of a film projector.

[2]The images may also fail to agree for reasons not associated with the architecture itself: if there are errors in image alignment, uneven exposure between the photographs (due to different camera settings or lens vignetting), or if the pictures were taken under different lighting conditions. Also, one image may view the area to be rendered at a higher magnification than another, which will cause the resampled images to differ in their spatial frequency content. For purposes of this discussion we assume that the photographs are properly aligned, evenly exposed under similar lighting, and that each image views the area to be rendered with adequate spatial resolution to produce the desired novel view. These assumptions are most easily met when high-resolution images are taken with a calibrated camera during a single session.

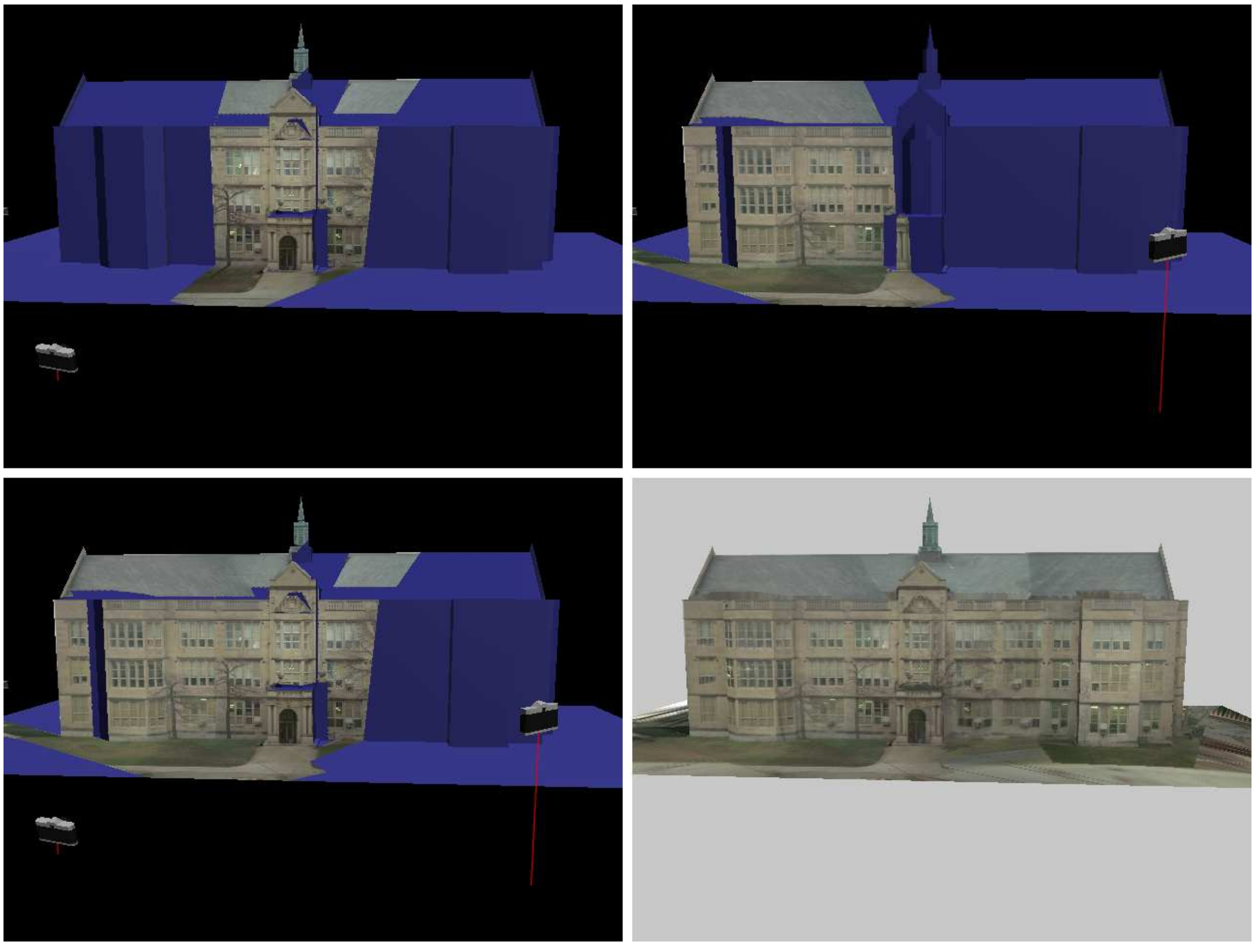


Figure 12: The process of assembling projected images to form a composite rendering. The top two pictures show two images projected onto the model from their respective recovered camera positions. The lower left picture shows the results of compositing these two renderings using our view-dependent weighting 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 [41].

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. 13.

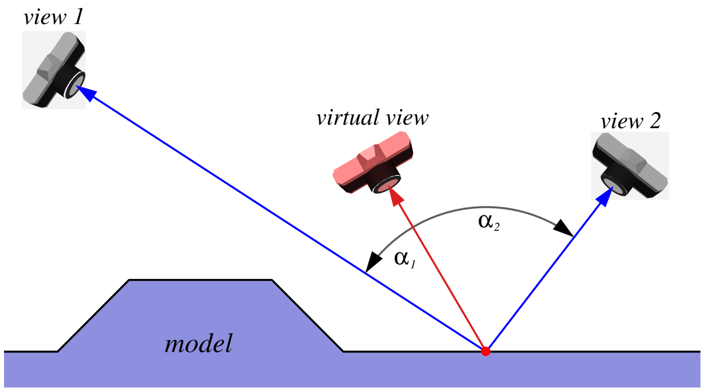
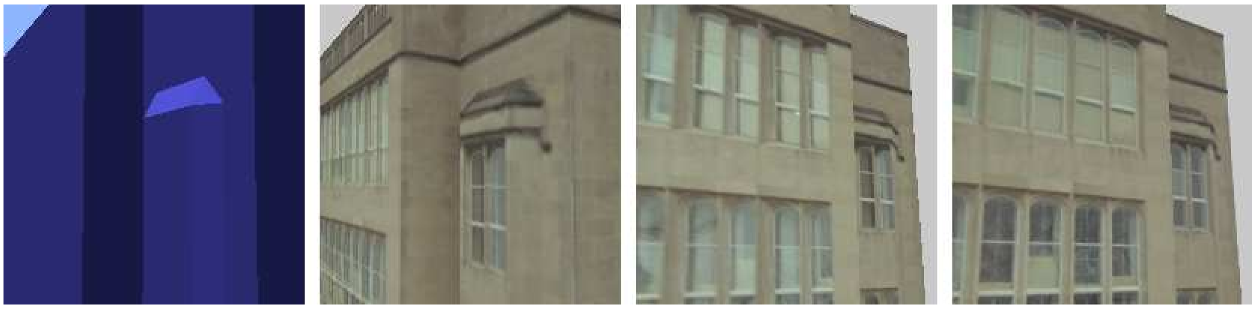


Figure 13: The weighting function used in view-dependent texture mapping. The pixel in the virtual view corresponding to the point on the model is assigned a weighted average of the corresponding pixels in actual views 1 and 2. The weights w_1 and w_2 are inversely inversely proportional to the magnitude of angles a_1 and a_2. Alternately, more sophisticated weighting functions based on expected foreshortening and image resampling can be used.

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 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 errantly 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 [41].

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,



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

Figure 14: 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.

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 [34].

6 Detail Recovery with Model-Based Stereo

The modeling system described in Section 4 allows the user to create a basic model of a scene, but in general the scene will have additional geometric detail not captured in the model. In our approach, the additional geometric detail is recovered automatically from the images with a *model-based* stereo algorithm. Unlike a traditional stereo algorithm, our model-based stereo algorithm measures deviations of the scene from an approximate model. This model serves to place the images into an common frame of reference that makes the stereo depth recovery much more effective. Like a traditional stereo algorithm, given two images (labeled the *key* and *offset*), our stereo algorithm computes a depth map for the scene from the point of view of the key image.

As with any stereo algorithm, our algorithm works by determining correspondences between points in the key and offset images that are projections of the same point in the scene. Given a point in the key image, a *correlation-based* stereo algorithm attempts to determine the corresponding point in the offset image by comparing (using some form of correlation) small pixel neighborhoods around the points. However, using correlation as the only factor in finding correspondences is rarely successful because of foreshortening, occlusions, and lack of texture as outlined in Section 2.2. Because of this, most stereo algorithms also use additional contraints, such as the epipolar constraint, the ordering constraint, and piecewise continuity:

The epipolar constraint. For every point P in the scene there is an *epipolar plane* which passes through P and the centers of the key and offset cameras. This epipolar plane intersects the two camera image planes in *epipolar lines* as shown in Fig. 15. From the figure, it can be seen that corresponding points must always lie along corresponding epipolar lines in the two images. Thus, when the camera positions are known, the stereo algorithm only needs to search along a point's corresponding epipolar line to find the point's correspondence. This reduces the search for correspondences from two dimensions to just one. We will show how our model-based stereo approach can take advantage of this constraint as conveniently as a traditional stereo algorithm.

The ordering constraint. This constraint enforces the observation that corresponding points usually appear in the same order on the epipolar lines, that is, the disparity of points along an epipolar line is monotone. The advantage of this assumption is that it often helps the stereo alogorithm avoid bogus correspondences. In addition, it can be implemented effciently using a dynamic programming technique [3]. The disadvantage is that this constraint is violated in situtations where a sufficiently narrow object close to the camera, such as a pole, is observed in front of a larger object further away. In our case, the modeling process usually breaks the scene into surfaces that obey the ordering constraint. When this is the case, the warping step of our model-based stereo algorithm generates images that obey the ordering constraint.

Piecewise continuity. For the typical image, a correct depth map consists of regions of continuously varying depth (corresponding to coherent objects) separated by depth discontinuities (corresponding to object boundaries). A stereo algorithm can exploit this constraint of piecewise continuity by requiring that neighboring pixels have similar depths, except across object boundaries. This constraint applies particularly well to the regular geometry characteristic of architectural scenes.

There are a number of obstacles that a stereo algorithm must overcome in order to produce meaningful disparity values; some of these obstacles are discussed in Section 2.2. For our application, the key and offset images will typically be taken from widely varying viewpoints relative to the depth of objects in the scene, which is a particularly difficult case for current stereo approaches for several reasons. First, the disparities between the key and offset images can be large which means that the stereo algorithm must investigate a large number of potential matches for each pixel in the key image. Second, corresponding regions in the two images will typically be viewed from different directions and thus exhibit varying degrees of foreshortening. This poses a significant difficulty for simple window-based correlation schemes since corresponding image neighborhoods will be scaled differently. Third, the two images may have very different patterns of occlusion — the entire side of a building may be visible in one image but not the other. Again, such a significant difference between the images is difficult for a traditional stereo algorithm to handle.

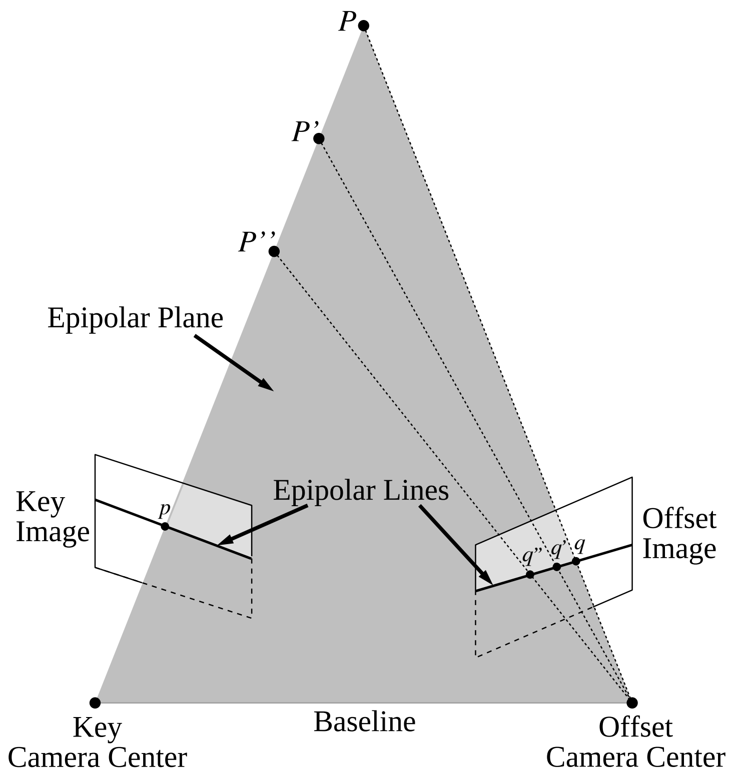


Figure 15: The epipolar geometry of a stereo pair. A point in an image, p, and the two camera centers determine an epipolar plane in space. The intersection of this plane with either image plane forms an epipolar line. When a point in the scene is observed in just the key image at p, it could actually be located at P, P', P'', or any other point along the same ray. Clearly, P, P', and P'' all project to p's epipolar line in the offset image at q, q', and q''. Thus, the correct match for p in the offset image must lie along the offset's epipolar line corresponding to p.

We overcome these problems by taking advantage of a coarse geometric model of the scene of the sort produced by the interactive technique of Section 4. Neighborhoods in the key and offset images are not compared directly; instead, the model is used to pre-warp the offset image into a new image, called the *warped offset image*, which has the following significant properties:

1. Any region in the scene which lies on the coarse geometric model will have zero disparity between the key image and the warped offset image.

2. Disparity between the images depends only on the difference between the scene and the model.

3. Differences in foreshortening between regions in the key and warped offset images are substantially reduced.

4. Any occlusions present in the model will be indicated in the warped offset image.

5. A simple linear epipolar geometry is maintained despite the warping. In fact, the epipolar lines of the warped offset image are simply the epipolar lines of the key image.

Most importantly, this warping is easily accomplished by the following method: the offset image is projected onto the model and rendered from the point of view of the key image. Fig. 16 illustrates this warping with a façade that has been coarsely modeled as a flat surface. The key and warped offset images appear similar—far more similar than the key and offset images appear. The differences between the key and warped offset image (which will be quantified by the stereo algorithm) are due entirely to how the actual façade deviates from the coarse model. Warping with respect to a single plane (as opposed to an entire model) has also been used to simplify image motion analysis in [27].

6.1 Establishing Stereo Correspondences

Once the offset image is warped, the stereo algorithm is ready to begin comparing pixel neighborhoods in order to establish stereo correspondences. However, the stereo algorithm needs to know, for any pixel in the key image, which pixels in the warped offset are geometrically possible matches. In traditional stereo, the set of possible matches is indicated by the offset image's corresponding epipolar line, as in Fig. 15. However, the warped offset image is nonuniformly projected and reprojected in a way that clearly does not preserve the same epipolar geometry. Because of this, we investigated exactly what sort of epipolar geometry exists after the warping step. We were pleased to discover that the epipolar geometry remains linear, regardless of the shape of the model. In fact, we discovered that the epipoles of the warped offset image are identical to the epipoles of the original key image. This result can be established mathematically as follows:

Proposition 1 *Let e_k represent the epipole, the image of the center of projection of the offset camera in the key image. The disparity between a point in the key Image, p_k and its correspondent in the warped offset image, q_k is directed towards the epipole in the key image, e_k.*

(a) *Key Image* (b) *Warped Offset Image* (c) *Offset Image*

Figure 16: (a) and (c) are two pictures of the entrance to the Peterhouse college chapel in Cambridge, England. The Façade system was used to model the façade as a flat surface and to recover the relative camera positions. In (b), the warped offset image is produced by using the offset image to texture map the geometric model as viewed from the position of the key image. In our model-based stereo approach, correspondence is performed between the key and warped offset images instead of the key and offset images. This is advantageous because the key and warped offset images are typically very similar in appearance; any differences are due to the unmodeled geometric detail.

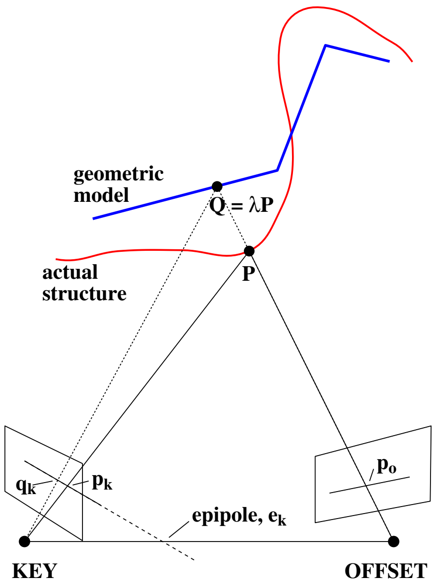


Figure 17: The epipolar geometry for model-based stereo.

Proof:

Fig. 17 shows a representation of an epipolar plane in a typical stereo configuration. The center of projection of the offset camera defines the origin of our coordinate frame of reference. $\mathbf{P} \in \Re^3$ represents the coordinates of a point in the world, $p_k \in P^2$ represents the projection of that point into the key image.

The 3x4 projection matrix M_k which defines how points in the world are mapped into the key image can be written as follows:

$$M_k = A_k(R\ \mathbf{T}) \tag{10}$$

Where $A_k \in \Re^{3\times3}$, $R \in SO(3)$ and $\mathbf{T} \in \Re^3$. The projection of the point $\mathbf{P}$ onto the key image, p_k can then be computed from the following expression.

$$p_k = M_k \begin{pmatrix} \mathbf{P} \\ 1 \end{pmatrix} = A_k R\mathbf{P} + A_k\mathbf{T}. \tag{11}$$

The epipole in the key image e_k is defined as the image of the center of projection of the offset camera in the key image.

$$e_k = M_k \begin{pmatrix} \mathbf{0} \\ 1 \end{pmatrix} = A_k\mathbf{T}. \tag{12}$$

The point $\mathbf{Q}$ represents the projection of point $\mathbf{P}$ onto the geometric model, that is $\mathbf{Q} = \lambda\mathbf{P}$ for some $\lambda \in \Re$; q_k is the image of the point $\mathbf{Q}$ on the warped offset image.

$$q_k = M_k \begin{pmatrix} \mathbf{Q} \\ 1 \end{pmatrix} = \lambda A_k R\mathbf{P} + A_k\mathbf{T} = \lambda p_k + (1-\lambda)e_k. \tag{13}$$

From equations (11), (12) and (13):

$$(e_k \times p_k) \cdot q_k = (e_k \times p_k) \cdot (\lambda p_k + (1-\lambda)e_k) = 0 \tag{14}$$

Equation (14) proves that p_k, q_k and e_k are collinear which implies that the disparity between a point in the key Image, p_k and its correspondent in the warped offset image, q_k is directed towards the *epipole* in the key image, e_k. $\square$

Effectively, Proposition 1 proves that the image warping process takes the points on an epipolar line in the offset image and maps them onto the corresponding epipolar line in the key image. The stereo algorithm that we have developed makes use of this fact by dividing the key image into a series of epipolar lines and determining correspondences along each of these lines independently [10, 20]. The similarity of regions in the key and warped offset images is evaluated using normalized correlation, and a dynamic programming approach is used to enforce the ordering constraint on the resulting disparity map [3]. After a disparity map has been computed for the epipolar line, a smoothing procedure is invoked which attempts to merge neighboring regions with similar disparities in order to produce a smoother disparity map. This process also fills in half-occluded regions in the key image with appropriate disparity values. Fig. 18 shows the disparity map produced for the key image in Fig. 16.

6.2 Stereo Results and Rerendering

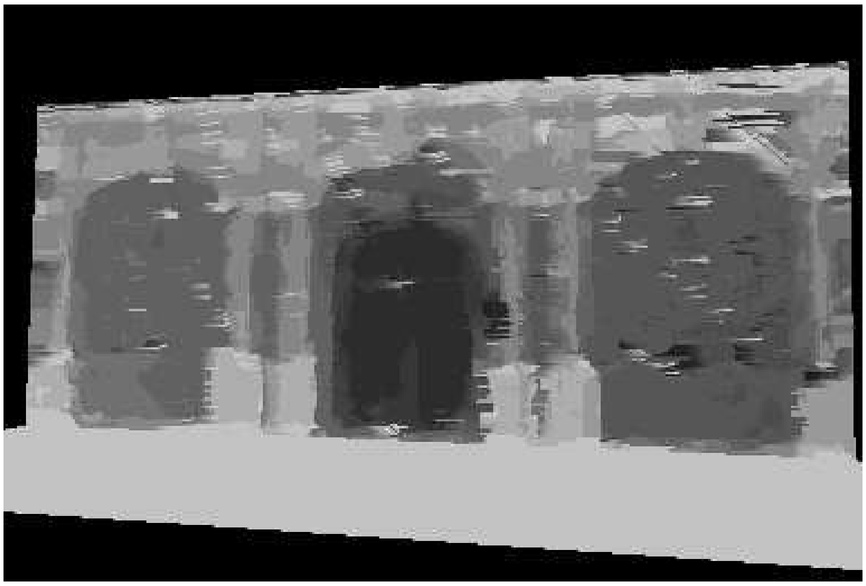


Figure 18: An unretouched disparity map produced for the image in Fig. 16(a), using our model-based stereo algorithm. In this image, points determined to lie closer to the front of the face are shown in lighter shades. For nearly all of the points, the depth appears to be computed accurately. In comparison, we found that naive correlation-based approaches were unable to compute reasonable depth estimates for more than half of the points.

Once a depth map has been computed for a particular image, we can rerender the scene from novel viewpoints using the methods described in [41, 31, 22]. Furthermore, when several images and their corresponding depth maps are available, we use the weighting function developed in the view-dependent texture-mapping method of section 5 to composite multiple renderings. The images in Fig. 19 were produced through such compositing.

Figure 19: 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 renderings are made using image-based rendering combined with view-dependent texture-mapping.

7 Conclusion

There are a several improvements and extensions that can be made to our approach. First, surfaces of revolution (such as domes, columns, and minarets) represent an important component of architecture that should be handled more effectively in the Façade modeling system. (The copper flames atop the pinnacles in Fig. 2 are approximated by polyhedra with 48 faces each.) Fortunately, there has been much work ([12], [19] [43]) that presents methods of recovering such structures from photographs. Curved objects are also entirely consistent with our approach to recovering additional detail through model-based stereo: the image warping process and the epipolar constraint in Observation 1 are valid for any model geometry.

Second, our stereo algorithm 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 Bi-directional Reflectance Distribution Functions) of the surfaces in the scene, and updating the image-warping methods to render non-Lambertian materials.

To conclude, we have presented a new, photograph-based approach to modeling and rendering architectural scenes. Our approach, which combines both geometry-based and image-based modeling and rendering techniques, is built from two components that we have developed. The first component is an easy-to-use photogrammetric 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 deviates from the basic model. Our method makes use of view-dependent texture mapping, a method we developed for using multiple images to better simulate geometric detail on basic models. 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.

7.1 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 Carlo Sequin, Tim Hawkins, and David Forsyth for their valuable help in revising this report.

References

[1] Kurt Akeley. Realityengine graphics. In *SIGGRAPH '93*, pages 109–116, 1993.

[2] A. Appel. Some techniques for shading machine renderings of solids. In *Proceedings of the Spring Joint Computer Conference*, pages 37–45, 1968.

[3] 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.

[4] P. Besl. Active optical imaging sensors. In J. Sanz, editor, *Advances in Machine Vision: Architectures and Applications*. Springer Verlag, 1989.

[5] F. C. Crow. Shadow algorithms for computer graphics. In *SIGGRAPH '77*, pages 242–247, 1977.

[6] James L. Crowley, Patrick Stelmaszyk, Thomas Skordas, and Pierre Puget. Measurement and integration of 3-D structures by tracking edge lines. *International Journal of Computer Vision*, 8(1):29–52, July 1992.

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

[8] O.D. Faugeras, Q.-T. Luong, and S.J. Maybank. Camera self-calibration: theory and experiments. In *European Conference on Computer Vision*, pages 321–34, 1982.

[9] O.D. Faugeras and G. Toscani. The calibration problem for stereo. In *Proceedings IEEE CVPR 86*, pages 15–20, 1986.

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

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

[12] A. Gross and T. Boult. Recovery of generalized cylinders from a single intensity view. In *Proceedings of the Image Understanding Workshop*, pages 319–330, 1990.

[13] Paul S. Heckbert. Survey of texture mapping. *IEEE Computer Graphics and Applications*, 6(11):56–67, November 1986.

[14] H. Hoppe, T. DeRose, T. DUchamp, M. Halstead, H. Jin, J. McDonald, J. Schweitzer, and W. Stuetzle. Piecewise smooth surface reconstruction. In *ACM SIGGRAPH 94 Proc.*, pages 295–302, 1994.

[15] William Jepson, Robin Liggett, and Scott Friedman. An environment for real-time urban simulation. In *Proceedings of the Symposium on Interactive 3D Graphics*, pages 165–166, 1995.

[16] 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.

[17] Craig Kolb, Don Mitchell, and Pat Hanrahan. A realistic camera model for computer graphics. In *SIGGRAPH '95*, 1995.

[18] 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.

[19] J. Liu, J. Mundy, D. Forsyth, and C. Rothwell. Efficient recognition of rotationally symmetric surfaces and straight homogeneous generalized cylinders. In *Proc. IEEE Conf. on Comp. Vision and Patt. Recog.*, pages 123–128, 1993.

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

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

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

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

[24] Michael Naimark. *Displacements*. San Francisco Museum of Modern Art, 1984.

[25] H. K. Nishihara. Practical real-time imaging stereo matcher. *Optical Engineering*, 23(5):536–545, 1984.

[26] 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.

[27] Harpreet S. Sawhney. Simplifying motion and structure analysis using planar parallax and image warping. In *International Conference on Pattern Recognition*, 1994.

[28] H. Shum, M. Hebert, K. Ikeuchi, and R. Reddy. An integral approach to free-formed object modeling. *ICCV*, pages 870–875, 1995.

[29] Steven Smith. *Geometric Optimization Methods for Adaptive Filtering*. PhD thesis, Harvard University, Division of Applied Sciences, Cambridge MA, September 1993.

[30] M. Soucy and D. Lauendeau. Multi-resolution surface modeling from multiple range views. In *Proc. IEEE Computer Vision and Pattern Recognition*, pages 348–353, 1992.

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

[32] Camillo J. Taylor and David J. Kriegman. Minimization on the lie group so(3) and related manifolds. Technical Report 9405, Center for Systems Science, Dept. of Electrical Engineering, Yale University, New Haven, CT, April 1994.

[33] 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.

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

[35] 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.

[36] 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.

[37] Greg Turk and Marc Levoy. Zippered polygon meshes from range images. In *SIGGRAPH '94*, pages 311–318, 1994.

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

[39] Thierry Vieville and Olivier Faugeras. Feed-forward recovery of motion and structure from a sequence of 2d-lines matches. In *International Conference on Computer Vision*, page 517. IEEE, December 1990.

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

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

[42] Y.Chen and G. Medioni. Object modeling from multiple range images. *Image and Vision Computing*, 10(3):145–155, April 1992.

[43] 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.