Interactive 3D Architectural Modeling from Unordered Photo Collections

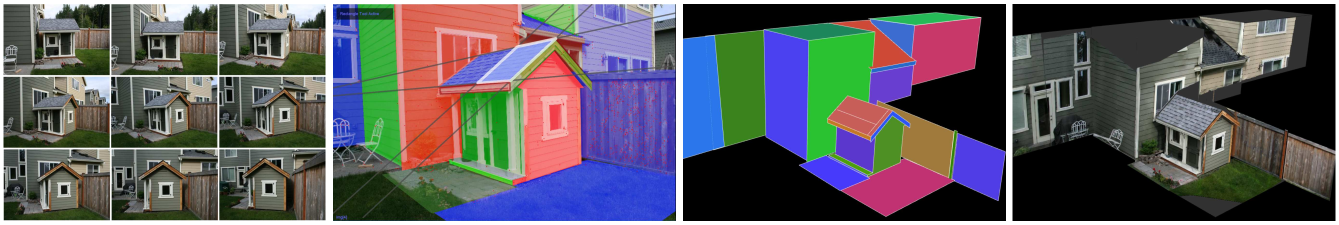
Sudipta N. Sinha
UNC Chapel Hill

Drew Steedly
Microsoft Live Labs

Richard Szeliski
Microsoft Research

Maneesh Agrawala
UC Berkeley

Marc Pollefeys
ETH Zurich and UNC Chapel Hill



(a) Input Photographs (b) 2D Sketching Interface (c) Geometric Model (d) Texture-mapped model

Figure 1: *Our interactive image-based modeling system provides an intuitive sketch-based interface for reconstructing a photorealistic textured piecewise planar 3D model of a building or architectural scene from an unordered collection of photographs.*

Abstract

We present an interactive system for generating photorealistic, textured, piecewise-planar 3D models of architectural structures and urban scenes from unordered sets of photographs. To reconstruct 3D geometry in our system, the user draws outlines overlaid on 2D photographs. The 3D structure is then automatically computed by combining the 2D interaction with the multi-view geometric information recovered by performing structure from motion analysis on the input photographs. We utilize vanishing point constraints at multiple stages during the reconstruction, which is particularly useful for architectural scenes where parallel lines are abundant. Our approach enables us to accurately model polygonal faces from 2D interactions in a single image. Our system also supports useful operations such as edge snapping and extrusions.

Seamless texture maps are automatically generated by combining multiple input photographs using graph cut optimization and Poisson blending. The user can add brush strokes as hints during the texture generation stage to remove artifacts caused by unmodeled geometric structures. We build models for a variety of architectural scenes from collections of up to about a hundred photographs.

CR Categories: I.3.7 [Computer Graphics]: Three-dimensional graphics and realism, image-based modeling, texture mapping—[I.2.10]: Artificial Intelligence—Vision and Scene Understanding

1 Introduction

In this paper, we develop a new interactive system for modeling architectural scenes from an unordered collection of photographs. We use computer vision techniques throughout to accelerate the modeling process. With our system, users can quickly generate a detailed textured 3D model of a scene. For example, the textured model in Figure 1 was generated in about five minutes.

Our interactive image-based modeling system uses an automatic feature matching and structure from motion preprocessing stage similar to [Snavely et al. 2006] to recover camera poses, a sparse 3D point cloud, and 2D feature correspondences between images. In addition, lines are extracted and used to automatically estimate vanishing points in the scene. The extracted 2D lines in each photograph are assigned vanishing directions that are consistent across the entire collection. The system then uses this geometric information at multiple stages of the interactive modeling process.

Once the automatic preprocessing is done, the user sketches 2D outlines of planar sections of the scene by drawing over photographs (please see the video). Not only can the user easily align polygon edges with vanishing point directions, but they can also directly draw 3D rectangles in a 2D image, using strokes that align with these directions (Figure 1b). The system estimates a 3D planar polygon from the user's 2D sketch using a robust estimation framework. Vanishing directions and nearby points from the 3D point cloud are used to estimate the plane normal and depth. A few such steps can quickly produce a piecewise-planar model of the scene.

Our system allows the user to visualize, edit and refine the geometry using an interactive 3D interface. As 3D geometry is created, the photographs are projected onto its surface, providing visual feedback during geometric editing. This allows users to accelerate tasks even for parts of the scene only observed by a single image, such as when adjusting the boundary of a 3D plane by aligning it to the appropriate image edge. By tightly integrating a 3D editing environment in our system, the user experience remains largely the same for parts of the scene observed by many images as for parts observed by only one or even zero images.

In order to texture the model, our system generates texture maps using graph cut optimization and Poisson blending to compute seamless texture composites by combining patches from multiple input photographs. During this step, the user can optionally use a brush interface to specify which pixels from a source image should be favored and which ones should be avoided.

In our system, the tedious process of manually establishing correspondences between images in order to estimate camera poses and the difficult UI problem of specifying 3D geometry in a 2D interface are eliminated. Instead, users are able to focus their energy on problems that are hard for the computer, such as segmenting the scene into planar polygons and choosing the desired level of detail.

1.1 Related Work

The problem of 3D modeling from images and video has received tremendous interest in the computer graphics and vision communities. Significant success has been recently reported with both fully automated systems such as Pollefeys et. al. [2004; 2008], Goesele et. al. [2007] as well as interactive systems such as Debevec et. al. [1996], Cipolla et. al. [1999], Oh et. al. [2001], El-Hakim et. al. [2005] and van den Hengel et. al. [2006; 2007]. Automated systems based on structure from motion process image sequences to first recover the camera poses and a sparse (point cloud) reconstruction of the scene. From the sparse reconstruction, dense multi-view stereo algorithms can generate a dense mesh model. While systems such as Pollefeys et. al. [2004; 2008] were geared toward processing video, recently Snavely et. al. [2006] used improved feature extraction and matching techniques to make structure from motion work with unordered photo collections obtained from the internet. This approach has led to an intuitive photo navigation system called Photo Tourism and allowed Goesele et. al. [2007] to compute dense mesh models using multi-view stereo. On the other hand, Dick et. al. [2004] showed that a probabilistic model-based method with appropriate priors could also be used to reconstruct buildings.

While these results are impressive, they require dense photo collections and their quality tends to suffer if either the camera motion is degenerate or the scenes lack adequate textures. These limitations can be overcome by having a user in the loop to interactively guide the geometry creation. Façade [Debevec et al. 1996] was one of the earliest image-based modeling systems designed for modeling architectural scenes which later gave rise to a commercial product called Canoma. It provides a set of parameterized 3D primitives such as cuboids, prisms, and pyramids. The user selects the appropriate primitive to model a part of the scene and then aligns it by pinning its vertices or edges to specific locations in the different photographs. These systems often require the user to manually specify correspondences of geometric primitives in multiple photographs although single-view reconstruction is sometimes possible in special cases using symmetry or vanishing point constraints. Adding many photographs can be quite laborious in these systems, so only a few well-planned photographs are typically used.

Instead of using a set of pre-defined shapes [El-Hakim et al. 2005] proposed a user guided method for creating and re-using building blocks for adding in geometric detail once a coarse model has been generated. Single-view modeling techniques such as Criminisi et. al. [2000], Oh et. al [2001] and other methods such as [Cipolla and Robertson 1999; Wilczkowiak et al. 2005] have used vanishing point constraints in modeling architecture. However all of these previous techniques require more manual intervention than our technique, are restricted to the use of solid primitives (e.g., parallelepipeds), and cannot easily exploit the large number of images commonly available today for performing 3D reconstructions.

Recently, van den Hengel et. al. [2007] proposed a system called VideoTrace to interactively model geometry from video; it has a tracing interface and is capable of utilizing information recovered by structure from motion. In their system, the user traces polygons over the frames of video. The 2D user interaction is converted into 3D reconstructions using geometric information obtained by applying structure from motion to the video. Corrections to the reconstructed surface can be made by moving a few frames forward or backward in the video and modifying the vertex positions.

Like VideoTrace, our system takes advantage of the underlying sparse reconstruction to infer the 3D geometry intended by the user when sketching on an image. Our system, however, also extracts and makes use of vanishing directions, which we have found to be a powerful tool for 2D-to-3D modeling applications. This allows us to accurately reconstruct polygonal faces by sketching in a single image without needing refinement in other images to obtain a globally consistent model. This makes our system more suited for buildings and architectural scenes while Videotrace would be better for free-form shapes. It also seems that the Videotrace interface would be more effective when camera baselines are small (as in video). In Videotrace, the user draws on a video frame but subsequently needs to do more image-based interactions (e.g. constraining the polygon vertices) after moving to a nearby frame in the sequence. The modeling time and effort in our system depends on the level of detail in the desired model, not the number of photos.

Finally, once the scene geometry is created, our system performs image-based texture map generation by building upon advances in panoramic stitching and image blending [Pérez et al. 2003; Agarwala et al. 2004]. We use a graph cut based optimization framework similar to that used by [Lempitsky and Ivanov 2007] – we have also incorporated a stroke-based editing tool into the framework, to allow users to edit the image-based texture maps for removing artifacts due to unmodeled structures such as trees and foliage.

Sketch-based modeling interfaces are another inspiration for our system [Zeleznik et al. 1996; Igarashi and Hughes 2001]. These interfaces allow users to quickly create 3D models from simple 2D drawings and gestures. Recent commercial systems such as SketchUp have been designed to support such sketch-based modeling. Yet, none of these systems incorporate techniques from image-based modeling. As a result they cannot easily produce the kinds of texture-mapped models our system is capable of. A new version of SketchUp now includes a *PhotoMatch* feature, which allows users to sketch over a photograph and model its geometry. This requires the user to carefully specify a scene coordinate system (three vanishing points) in each input photograph. Unfortunately, many close-up photographs (not taken at corners of buildings) may not have enough information to reliably indicate such triplets of vanishing points (converging parallel lines). In contrast to systems like Facade [1996], Sketchup and Videotrace [2007] our system can process large unordered photo collections and avoids the need for carefully taken photos. The extra redundancy in the set of input photographs allows our system to synthesize seamless high-resolution texture maps without artifacts due to unmodeled occluders.

While some tools in our system such as sketching polygons and snapping to vanishing directions have roots in sketch-based modeling systems, we also provide a continuum of modeling modes, from purely CAD based modeling to computer vision aided modeling and texturing. We demonstrate this flexibility by implementing a few standard CAD utilities (extrusion, plane completion, mirroring about axis aligned planes etc). The system is designed so that it can fall back to traditional CAD-style modeling when parts of the model or scene are observed by only few or even zero images.

2 Preprocessing

Although our system is interactive, it relies on accurate knowledge of the camera orientation and sparse 3D geometry estimated from the observed scene. Our system starts by pre-processing all the photographs using computer vision techniques for (a) reconstructing sparse 3D structure from images and subsequently (b) for estimating vanishing points consistently across multiple views.

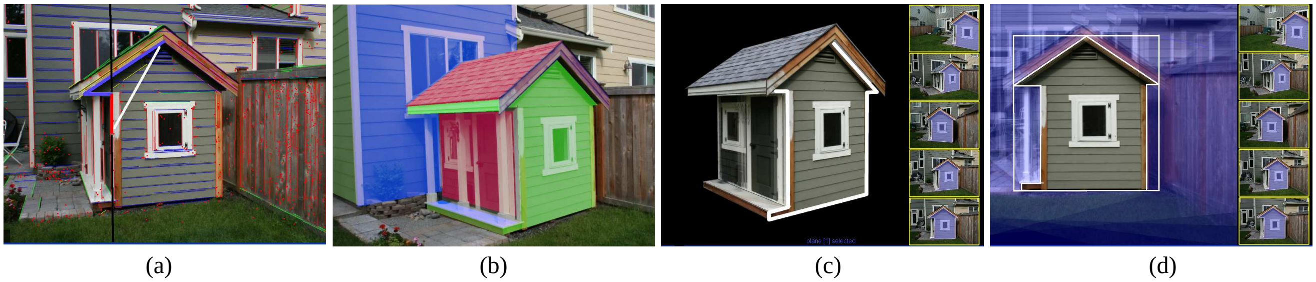


Figure 2: *Screenshots of our system in use. (a) Photo browser with reprojected 3D points and vanishing line clusters displayed. The user sketches over the photograph in this mode. (b) Photo browser with overlaid translucent partial 3D model. (c) The interactive 3D viewer, with the side panel displaying automatically selected views for the chosen plane. (d) An orthogonal view of the chosen plane.*

2.1 Structure from motion

We compute the camera positions and a sparse 3D point cloud using an approach similar to the one proposed by Brown and Lowe [2005]. Features are detected in all images, matched across image pairs, then used to robustly estimate the camera poses and the position of points in the sparse 3D point cloud in an incremental fashion [Brown and Lowe 2005; Snavely et al. 2006]. The optimal camera and point parameters are ones that minimize the error between the reprojected 3D points and the detected 2D interest points. This minimization is accomplished by solving a sparse, iterative, non-linear least squares problem referred to as *bundle adjustment* [Triggs et al. 2000]. In addition to the camera poses and 3D point positions, the list of cameras that observe each point is retained and used by our system to propagate user interactions from the reference view to other views.

2.2 Multi-view Vanishing Point Extraction

Parallel lines are common in architectural scenes containing man-made structures. Under perspective projection, parallel lines appear to meet at a point in the image called the *vanishing point* (VP). Vanishing points have been extensively studied along with the geometry of image formation and have been found useful for camera calibration and 3D reconstruction from a single uncalibrated image [Criminisi et al. 2000]. Although automatic methods for robust and stable VP estimation in a single image such as [Rother 2002] are well known, little work has been done on jointly estimating them in multiple images of the same scene.

During the pre-processing stage, we automatically extract up to three orthogonal VPs in each image, one of which always corresponds to the vertical direction in the scene. VPs are first independently extracted from each image, and then these are jointly optimized to align the vanishing directions and make them globally consistent across all the images. Additional VPs can be selected interactively from a set of candidate VPs or by drawing a pair of lines in a single image which are known to be parallel in the 3D scene.

The details of the optimization approach for simultaneous vanishing point estimation and camera pose refinement can be found in Appendix A. While this is not the main focus of the paper, we have not seen any previous work on structure from motion that performs this additional optimization. In previous work, vanishing points were mostly used within a single-view reconstruction framework. Such systems recovered VPs automatically from single views and often required strategic views such as the corner view of a building. In our system, the multi-view VP estimation step is more accurate as we jointly estimate the vanishing directions in the scene. Having globally consistent VPs across multiple views allows the user to sketch on any image of his choice in the photo collection.

3 Geometric Reconstruction

In this section, we describe the front end of our system and the details of the user interaction. There are two basic modes of operation: a 2D photo browser with a sketch-based drawing interface (Figures 2a–b and 3) and a 3D editor with a set of interaction tools for reconstruction and texture map generation (Figure 2c–d).

3.1 2D Photo Browser and Sketching Interface

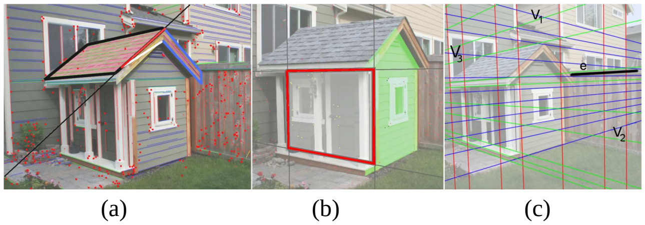
The 2D photo browser allows the user to navigate through the photo collection and select appropriate images to sketch over. As shown in Figure 2a, the user can view both the 2D interest points corresponding to the reconstructed 3D points on the image (*reconstructed interest points*) and the family of lines corresponding to different vanishing points. Each family of parallel lines is drawn using the same color in all the images. As a 3D model is built up, it can be rendered from the camera viewpoints and translucently displayed over the corresponding photographs with a separate color for each planar facet (Figure 2b). This helps the user determine which parts of the reconstructed scene are still incomplete.

The sketching interface in this mode includes a polygon tool, which allows the user to trace out closed 2D polygonal outlines over the photograph (Figure 3a), as well as a rectangle tool, which lets the user directly draw a 3D rectangle aligned with vanishing point directions (Figure 3b).

By tracing outlines on the image, the user is identifying a planar facet in the scene. Once the polygon or rectangle has been drawn, the system estimates the parameters of the plane (normal direction and depth) that best fit the 3D points enclosed by the polygon. The polygon is then simply projected from the image onto the estimated plane to produce the 3D planar segment. While sketching on the image is often a natural and intuitive way to sketch, it also circumvents the need for the system to try to estimate the boundary curve automatically, which is often a challenge for fully automated approaches. In order to make the drawing process easier and the 2D-to-3D inference more robust, our user interface provides two forms of snapping: (a) snapping to VP directions and (b) snapping to pre-existing geometry.

VP snapping. If any of the line segments drawn by the user almost passes through a VP, that line segment is snapped to exactly pass through it. Snapped line segments are constrained by the system to be parallel to one of the detected vanishing directions (Figure 3b). The VP snapping feature is enabled by default, but can be easily disabled when necessary.

By snapping edges to one or more vanishing points, the user provides powerful constraints to the system for estimating the plane pa-



(a) (b) (c)

Figure 3: *(a–b) Screenshots showing the user traced polygonal outlines with snapping to previous geometry and vanishing points. (c) VP Ambiguity: The user drawn edge such as e can get associated with multiple VPs during snapping i.e. V_2 or V_3 in this case. The system remembers both possibilities and chooses the correct one during the RANSAC-based plane fit step.*

rameters. Without any vanishing point constraints, the system must estimate all three parameters of the plane, two for the normal direction and one for its depth. A single vanishing point constrains the normal direction of the plane, reducing the plane estimation problem to a two parameter estimation problem. If the user specifies two vanishing points, the plane normal is completely constrained, and the system only needs to estimate one parameter–the depth of the plane.

While using VP snapping, ambiguities arise when the line segment drawn by the user is collinear with two or more vanishing points in the image (Figure 3c). In this case, during the robust plane reconstruction step described in Section 3.2, the system automatically chooses the VP that produces a plane which fits the point cloud better.

Edge snapping. The second form of snapping is useful when the user desires to reconstruct multiple planar facets that share edges, e.g., two adjoining façades of a building, as shown in Figure 3a. While drawing the polygonal chain, if a new line segment is close to being collinear and also overlaps with the reprojection of a previously drawn 3D edge, we force the user-drawn line segment to be exactly collinear (in 3D) with the previous edge. This also ensures that the new planar polygon that gets created will be welded to the previously generated plane. However the plane equations of the two planar facets involved are not affected; only their polygonal boundaries are transformed appropriately. Edge snapping is disabled by default, but when a candidate edge is available for snapping the corresponding polygon changes color indicating to the user the possibility to perform snapping.

3.2 Constrained Plane Fitting

To reconstruct a planar segment, the user draws its polygonal outline P in a selected image I. Its backprojection from the camera center C is a cone in 3D, and any planar cross-section of this cone is a potential plane candidate. A robust RANSAC [Fischler and Bolles 1981] based plane fit on the 3D point cloud is then performed to find the best plane that satisfies the user's input.

First we find X_1, the subset of *reconstructed 2D interest points* originally detected in image I that lie within the boundary of the polygon P drawn by the user in that image (see Figure 4). Each 3D point in X_1 is linked to reconstructed 2D interest points in other views as well. We compute the set union of such views for all points in X_1 and denote the set of corresponding cameras by $\{J_i\}$. Next, we find the set of 3D points that project within the boundary of P in image I and are also linked to reconstructed 2D interest points in at least two views included in $\{J_i\}$. Let us denote this set by X_2. (Note that $X_1 \subseteq X_2$.) RANSAC is now performed using the set X_1 for generating hypotheses and X_2 for evaluating them. This ensures that the plane fit is not overly sensitive to the choice of the selected image.

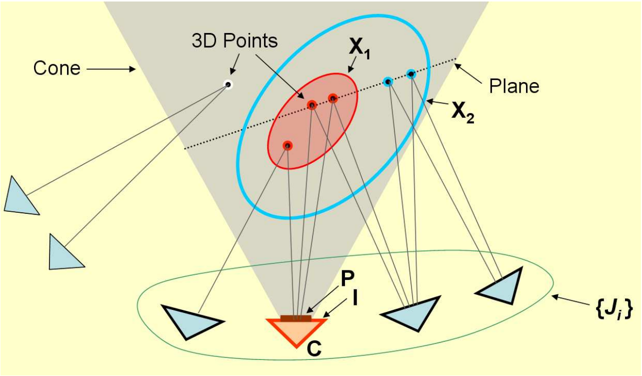
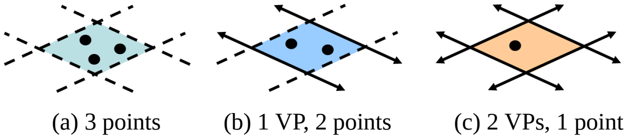


Figure 4: *Plane fitting: A 2D slice of the 3D scene is shown. Polygon P is shown by a dark line in the selected image I. Planar cross sections of the cone (shown in gray) defined by the camera center C and the user drawn polygon P in image I are evaluated as potential candidates for the plane fit using a RANSAC based approach.*



(a) 3 points (b) 1 VP, 2 points (c) 2 VPs, 1 point

Figure 5: *Multiple models for oriented planes used in RANSAC based plane-fitting. Solid lines depict vanishing point constraints.*

To robustly estimate the plane parameters, a RANSAC [Fischler and Bolles 1981] framework is used. Many random samples of points are drawn from the selected subset. These samples are used to generate plane hypotheses, which are then scored against all the points in the selected subset. If no vanishing point constraints are available, three non-degenerate points, i.e. not collinear or coincident are needed in each sample to compute a plane hypothesis (Figure 5a). With one vanishing point constraint, only two points need to be sampled (Figure 5b). Finally, with two vanishing point constraints only one point is needed (Figure 5c). Each plane hypothesis is evaluated by computing the (robust) re-projection error of each 3D point with respect to its closest point on the plane. The best hypothesis is then polished using (robustified) constrained least-squares.

3.3 3D Viewer and Editor

Our interactive 3D viewer supports different rendering modes and standard viewpoint controls, acts as a geometric editor with specific capabilities, and also provides a front end for the image-based texture mapping engine. The viewer allows the user to interactively select a particular planar segment either to edit its geometry or texture map. The 3D model being built can be rendered either (a) with projective textures or (b) with the texture maps created by the system. Blended projective texturing provides a way to visualize out-of-plane parallax and gives the user visual feedback at interactive rates to guide the geometric editing. Whenever the 3D model is well aligned with the photographs, the projected textures are geometrically well aligned on the planar facets of the model, and their blended (average) value therefore looks sharp.

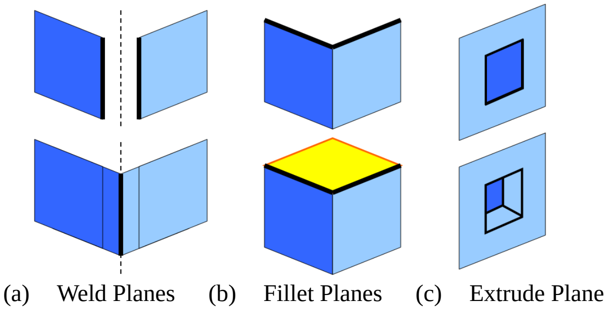


Figure 6: *3D editing operations: (a) welding (extending) two planes up to their intersection edge; (b) creating a fillet plane from two existing line segments; (c) using extrusions to create recessed windows and doors.*

The user can switch the viewpoint in the 3D viewer to specific configurations, such as a camera viewpoint or an orthographic view of the selected plane. A side panel in the viewer displays thumbnails of a subset of photos in which the selected plane is observed. These are referred to as the *selected views* for the currently active plane. (Section 3.4 describes how these are computed.) The selected plane is also rendered from these viewpoints and displayed over the thumbnails as a translucent color overlay (Figure 2c–d).

The following forms of user interaction can be performed from any suitable viewpoint in the interactive viewer.

Welding planes involves choosing a pair of disconnected planes and joining them along a *weld line* – the 3D line along which the two underlying planes intersect. Once the two planes are chosen, other planes are temporarily hidden to avoid clutter and the 3D weld line is displayed (Figure 6a). Individual polygon edges can then be selected and snapped to the weld line. Since each edge is individually snapped, partial welds are possible. This is useful where one of the two planes protrudes beyond the weld line, e.g., when a roof overhangs a wall. Both the fillet plane and extrusion tools (described below) generate planes that are automatically welded to adjoining geometry.

We treat each new weld as an additional constraint on the vertices involved in the weld, forcing them to belong to multiple planes (up to 3, since only 3 planes in general position intersect at a unique point). Thus, once planar patches are welded together, future transformations on the vertices involved are allowed only in a 0, 1 or 2 dimensional subspace. Vertex positions that respect the weld constraints are computed using Lagrange multipliers (Appendix B).

Editing plane parameters involves transforming the underlying plane equation of the selected planar segment. One or more of the following transformations can be applied in succession: (a) sweep the plane along its normal direction while keeping the normal fixed; (b) select a rotation point on the plane and rotate the plane about it in an arc-ball fashion; or (c) draw a line on the plane and rotate the plane about the line. These operations do not modify the shape of the planar segment unless it is welded to neighboring planes.

Editing the plane boundary involves modifying the polygonal outline, i.e., the shape of the planar segment. Its vertices or edges can be moved within the plane. Additional vertices can also be inserted into the polygonal chain. The degrees of freedom for this motion varies between 0 and 2 depending on the number of weld constraints on the vertices of the polygon. This operation can also perform snapping. When a line segment is nearly parallel to one of the families of parallel lines (corresponding to vanishing directions within this plane), snapping forces it to be exactly parallel.

Fillet planes can be created to connect pairs of existing planes as follows. The user selects a pair of non parallel line segments (the polygon edges of two different planar segments) and the system fits a parallelogram to the pair of line segments (Figure 6b). A least squares fit is performed to deal with non-coplanar line segments.

Extrusions can be created in various ways. An existing planar segment can be extruded along its normal direction to create a swept volume. Another form of extrusion involves drawing polygonal cutouts on an existing plane and then extruding the polygons inward (Figure 6c). This is useful for modeling structures such as windows, doors and other details in façades of buildings. To deal with repeated patterns often found on such façades, we provide the ability to replicate cutouts within the same plane and snap them into position for exact horizontal and vertical alignment. (See the accompanying video.)

3.4 View Selection

Every planar segment in our reconstructed model has a set of k (typically $k \leq 6$) *selected views* associated with them, which are the images in which the planar segment is expected to be significantly visible. We solve the visibility problem approximately by combining the following heuristics. The set of 3D points that belong to the reconstructed plane (i.e., those that formed the inlier set during the RANSAC-based plane fit) are each linked to interest points detected in specific photographs. The fact that these 2D interest points were matched across views and generated 3D points in the point cloud implies that the plane must have been visible in the images where they were detected.

Thus, based on 2D interest point count, we pick k cameras from the set of selected views. In the case of very few or no inliers, we choose k most frontal cameras after rejecting all cameras observing the plane at a grazing angle ($> 80^o$). The selected views are used in the RANSAC-based plane fit step, the projective texturing in the 3D viewer, and for generating the image-based texture maps.

4 Texture Mapping and Blending

To compute a texture map for each planar segment, we backproject the selected views onto the plane. As the reconstructed plane is typically visible in multiple images, pixels from each of these backprojected images provide potential candidates for the texels in the target texture map. The resolution of the target texture map is determined by choosing the coarsest grid that avoids aliasing in all of the selected images for the chosen plane. We perform visibility estimation on a per-texel basis to determine a set of visible pixel candidates. A simple way to obtain a texture map would be to blend together all the candidates for each texel, but this produces ghosting artifacts noticeable in the case of misalignment or unmodeled geometric detail and in general blurs the fine resolution present in the photographs. Choosing the pixel from the most frontal view for each texel independently would avoid these problems but would create other artifacts in the form of noticeable seams in the texture map. In addition, none of these approaches deal with the problem of partial occlusion of the modeled surface by foreground occluders.

We therefore pose texture map generation as an MRF optimization problem, where a high quality seamless texture map is computed by minimizing a suitable energy functional consisting of data penalty terms and pairwise terms. While Lempitsky and Ivanov [2007] used a similar technique for generating image-based texture maps, their underlying MRF was defined on a tesselated triangulated manifold mesh instead of a texel grid. We also incorporate user constraints specified in the form of brush strokes and use the sparse depth samples obtained from the 3D point cloud in our energy functional to create texture maps with fewer foreground occluders.

Our texture mapping stage proceeds as follows. The system automatically creates a texture map composite for each plane. This is done by performing an automatic graph cut optimization to optimally choose source images for the texture map composite. The user is then given the option to modify the result interactively by specifying constraints using brush strokes drawn on the reconstructed plane. Each interaction results in an additional iteration of the graph cut optimization on the same MRF. When the user is satisfied with the composite, Poisson blending [Pérez et al. 2003] is performed to reduce seam artifacts and this generates the final texture map.

4.1 Graph Cut Optimization

Graph cut optimization is the preferred technique used in computation photography for finding optimal seams between image regions being stitched together [Agarwala et al. 2004]. This technique minimizes the following energy function. We denote the set of aligned images rectified to the target plane by $I_1, \ldots, I_n$. The graph cut estimates a label image L where the label at pixel p denoted by $L(p)$ indicates which image I_k should be used as the source for p in the target texture map. The energy functional we minimize is denoted by $E(L)$ where L is a particular label image. Note that L is piecewise constant except at seams between adjacent pixels p and q, where $L(p) \neq L(q)$.

Our energy functional E (see Equation 1) is the sum of a data penalty term summed over all pixels of the label image L and a pairwise interaction penalty term summed over all pairs of neighboring pixels in L.

$$E(L) = \sum_p D(p, L(p)) + \sum_{p,q} S(p, q, L(p), L(q)) \qquad (1)$$

The data penalty term denoted by $D(p, L(p))$ stores the cost of assigning label $L(p)$ to pixel p while the interaction penalty term $S(p, q, L(p), L(q))$ stores the cost of assigning labels $L(p)$ and $L(q)$ to neighboring pixels p and q in the label image.

$$\begin{aligned} S(p, q, L(p), L(q)) &= 0 \quad if \;\; L(p) = L(q) \\ &= A + B \end{aligned}$$

$$\begin{aligned} A &= |I_{L(p)}(p) - I_{L(q)}(p)| + |I_{L(p)}(q) - I_{L(q)}(q)| \\ B &= |\nabla I_{L(p)}(p) - \nabla I_{L(q)}(p)| + |\nabla I_{L(p)}(q) - \nabla I_{L(q)}(q)| \end{aligned}$$

We use the same pairwise seam term as used in [Agarwala et al. 2004] to encourage invisible seam transitions. Minimizing the ratio of the average color difference and the average gradient difference generates invisible seams that are biased towards strong image edges. However this can lead to artifacts in the Poisson blending. Therefore we prefer seams such that colors and gradients are matched across it without the strong edge bias. The results of running our graph cut seam finding on a house wall are shown in Figure 7. Our data term is a weighted sum of the following four terms.

Preference for a frontal view. We use $D_1 = 1 - \cos^2(\theta)$, where θ is the angle between the plane normal and a particular camera.

Photo-consistency. We find the median color μ of the set of candidate pixels and set a high penalty for choosing a pixel whose color $I_{L(p)}(p)$ deviate from the median color. Specifically, we use $D_2 = |I_{L(p)}(p) - \mu|$.

Brush Strokes The user can specify additional constraints using brush strokes. Constraints can be both positive (the user draws in one of the source images indicating the preference for pixels from that image in the vicinity of the brush stroke) or negative (the user wants to erase pixels in the current estimate of the texture map and replace them with alternate pixels from other source images). An infinite penalty is used to enforce such constraints in a hard manner.

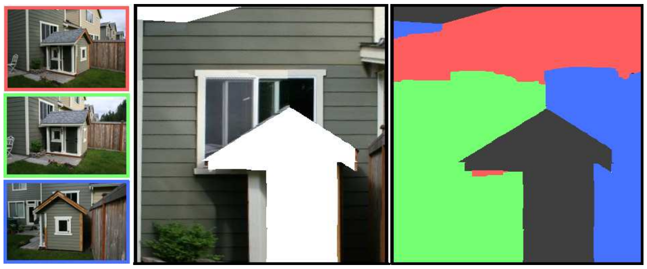


Figure 7: *(Left) Source images for building the texture map. (Middle) Texture map generated by the graph cut. (Right) Label map.*

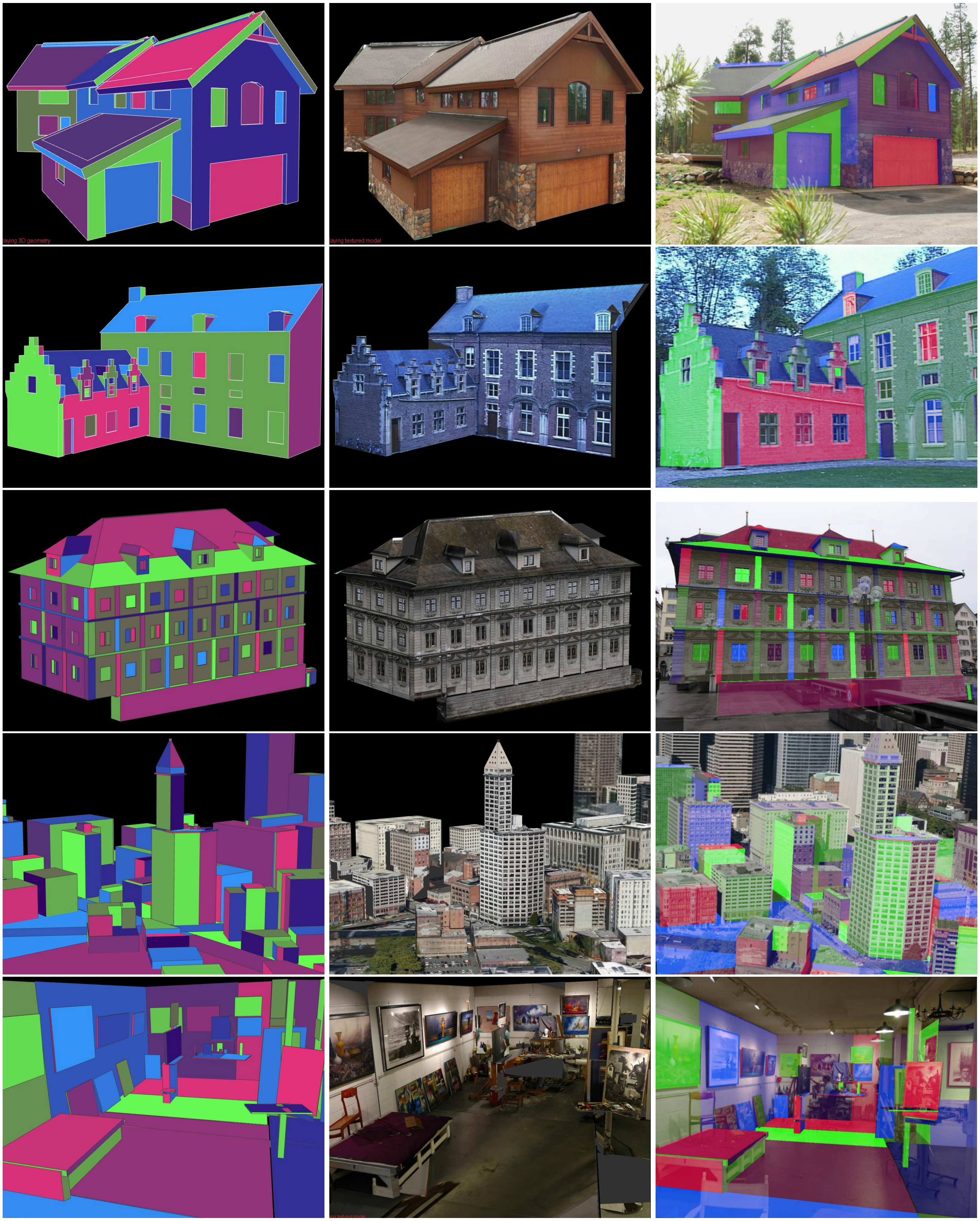
Sparse Point Cloud Depth By projecting the 3D point cloud into all the images, we obtain accurate depth at a few pixels. If a pixel with known depth corresponds to a 3D point on the target plane, then the corresponding image is a good candidate for texturing that pixel. A positive constraint for preferring this image as a source is introduced in a 3x3 patch around the pixel in the texture map. Otherwise the 3D point is part of a different geometric structure. If the distance between the 3D point and the nearest point on the plane when reprojected into a particular image exceeds a threshold (≥ 10 pixels) we introduce a negative constraint for choosing that image as a source for the corresponding pixel in the texture map.

5 Results

Using our system, we have reconstructed a variety of scenes ranging from small houses (Figure 1 and Figure 8(top)) to complex architecture, aerial urban footage and an artist gallery (Figure 8 (bottom)). Structure from motion worked accurately on all these sequences. While dealing with structure from motion failures is important, we do not address this issue here but will investigate it in future work. Tables 1– 3 report the number of photos used and the time spent by the user while building these models in our system. For all the datasets except for the city scene which contains many buildings, the user was able to build a coarse model of the whole scene very quickly.

In the Leuven castle sequence, very few lines associated with the vanishing directions of the two roofs were detected in the images. The system could reconstruct the roof planes accurately without requiring these oblique vanishing directions to be specified. For both the castle and the Zurich Rathaus dataset, we found the replication of planes and polygon cutouts aided by the horizontal/vertical alignment snapping particularly helpful for extruding the windows. The aerial city images contain parallel streets and offset city blocks at multiple orientations. Multiple vanishing directions proved to be extremely beneficial here.

The photographs of the brown house contain foreground occluders such as trees. Avoiding such clutter is sometimes impossible while photographing buildings and architecture while walking around them. For the trees in the brown house dataset that were observed in multiple images, the structure from motion preprocessing often found several points on them. Since these 3D points are not on the wall being textured, many of the trees are automatically eliminated from the texture map. In some cases, when the automatic graph-cut step could not eliminate the occluders completely,



(a) Geometric Model (b) Texture Mapped Model (c) Model overlaid on photo

Figure 8: *Results from five different photo collections shown from top to bottom - (1) Brown House (61 images); (2) Leuven Castle (28 images) (3) Zurich Rathaus (98 images) (4) City Scene (102 images) and (5) Indoor Gallery Scene (158 images) (better seen in color).*

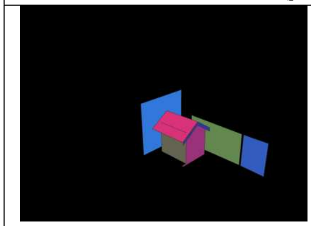
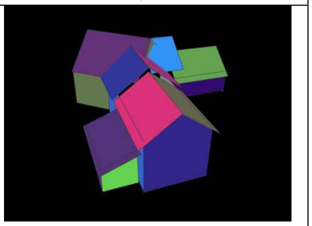
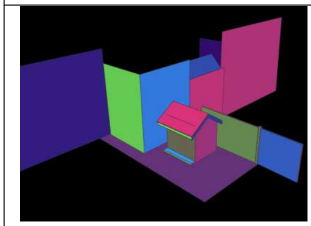
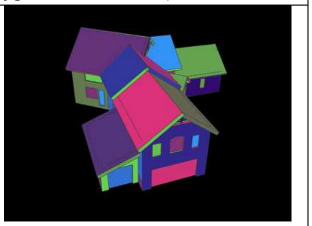
Playhouse (10 images)		BrownHouse (61 images)	
Coarse Models (polygon count, total time)			
			
8	2 mins	19	6 mins
Detailed Final Models (polygon count, total time)			
			
46	5 mins	311	25 mins

Table 1: *Polygon count (number of planar polygons) and total user time for the Playhouse and brown house models (better seen in color).*

the user specified a few brush strokes to remove them from the texture map. The final texture maps were almost fully free of artifacts due to un-modeled structures like the trees and bushes.

The advantage of being able to utilize hundreds of photos is evident in the artist gallery reconstruction. The close-up photos of the paintings and canvas boards allowed us to generate high resolution, photo-realistic texture maps for the 3D model. For the city dataset which had severe occlusions due to the presence of many high-rise buildings, using about a hundred photographs helped in generating more complete texture maps. Both severe occlusions and variation in lighting made texture map generation for this scene quite challenging. Sometimes our automatic view selection strategy did not provide maximal coverage for generating texture maps requiring the user to manually specify additional source images. This task will be fully automated in future and better strategies for dealing with lighting variation and auto-exposure will be investigated.

5.1 User Study

To evaluate the ease of use of our system, we conducted a user study with four novice users. After providing a 40 minute long demonstration and training, we asked the subjects to model the playhouse (Figure 1). We evaluated them on the playhouse by comparing the duration of their modeling sessions and the accuracy and level of detail of their results against those of an expert user. The subjects were able to create quite accurate and detailed models of the playhouse in 11 – 16 minutes while an experienced user took about 5 minutes. The new users were often satisfied with the automatically generated texture maps and did not edit the texture maps further, although a little editing would have removed the few artifacts that remained in some cases.

Finally one of the subjects was told to use the system to model a small scene of their own choice. The subject took 16 photos of a small shed and then successfully built a textured 3D model using our system (please see the video).

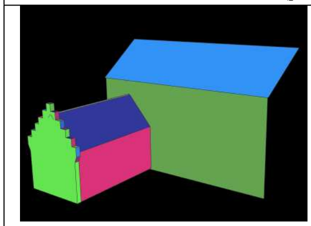
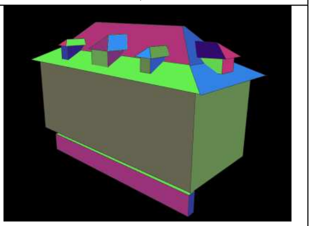
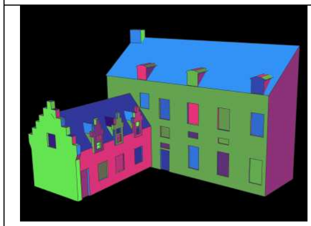
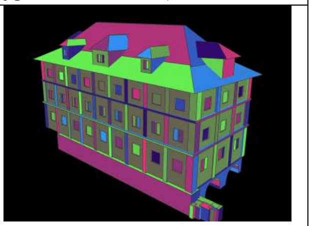
LeuvenCastle (28 images)		Zurich Rathaus (98 images)	
Coarse Models (polygon count, total time)			
			
35	4 mins	63	8 mins
Detailed Final Models (polygon count, total time)			
			
261	15 mins	616	30 mins

Table 2: *Polygon count (number of planar polygons) and total user time for the Leuven Castle and Zurich Rathaus models (better seen in color).*

5.2 Conclusions

Based on the user study, we found vanishing directions to be a powerful modeling aid for architectural scenes. Allowing users to snap line segments to vanishing directions not only makes the sketching interface easier to use, but also greatly enhances the system's ability to accurately estimate 3D planes. Using the vanishing point constraints during plane reconstruction proved critical for low texture surfaces such as the fascia boards lining the roof of the playhouse. This extra robustness greatly improves the user experience.

Users often had a tendency to sketch in regions where *reconstructed interest points* were absent. Currently our system is unable to generate a plane from the user's input in such cases. In the future, we will make the system more robust by generating denser point clouds using multi-view stereo and providing better visual feedback to guide the user while sketching.

New users found it intuitive to create disconnected planar polygons by sketching over the images followed by manual welding and plane editing in the 3D editor to generate connected surfaces. In some cases using the snap to pre-existing geometry feature while sketching would have been quicker but news users preferred the simpler but more time-consuming strategy.

Subjects found the visual feedback provided by projecting textures on the geometry in the 3D editor quite useful. They used this mode frequently for editing the plane boundary, creating extrusions, cutouts and for generating fillet planes.

Users also spent substantial time browsing the photo collection to select an appropriate photo to sketch over. Although this choice is not critical for the system, we believe that users will do this quite often to explore the photo collection in order to understand what can be reconstructed in the scene. To improve the user's experience, our system could be enhanced with a spatially aware photo navigation system similar to the one proposed by Snavely et. al. [2006]. The benefit of doing this would be evident while processing a large photo collection.

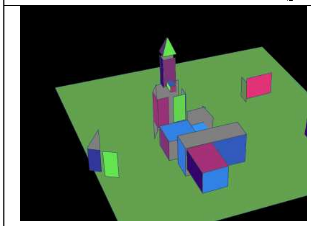
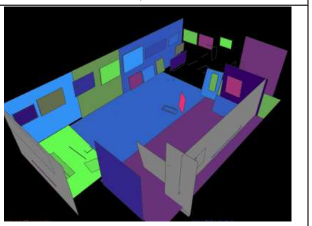
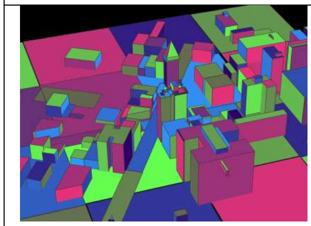
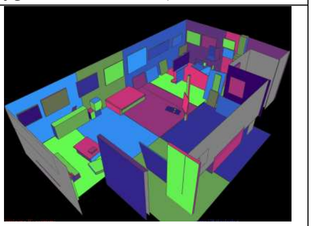
City Scene (102 images)		Gallery (158 images)	
Coarse Models (polygon count, total time)			
			
47	10 mins	38	10 mins
Detailed Final Models (polygon count, total time)			
			
746	2 hrs	568	2 hrs

Table 3: *Polygon count (number of planar polygons) and total user time for the City and Gallery models (better seen in color).*

6 Summary and Future Work

We have introduced an interactive image-based modeling system for architectural scenes that leverages recent advances in automatic computer vision techniques and sketch-based interfaces for 3D modeling. Our system can handle large photo collections and is well suited for modeling architectural scenes as it utilizes vanishing point constraints. It also supports the generation of seamless high-quality image-based texture maps for the 3D models by utilizing all the images that are available.

In the future, we will use advanced computer vision algorithms to make our system more automatic. In particular we will explore approaches for automatically suggesting candidate planes. We plan to improve the pre-processing pipeline by performing vanishing point extraction coupled with 3D line segment reconstruction as well as integrating quasi-dense stereo into our system to generate denser 3D point clouds. In-painting will be incorporated into the texture map generation pipeline in order to deal with the problem of severe occlusions and missing information which occurs when the surface is not visible in any of the source images.

In our system, there is currently no mechanism for correcting the image sequence for color balance or to handle lens vignetting. This is important for accurate image-based texture map generation especially when the photographs are taken with auto-exposure settings. We will address this in future by incorporating automatic algorithms for radiometric calibration from image sequences such as those proposed by Kim et. al. [2008] to neutralize the effects of auto-exposure and perform correction for vignetting.

In practice, the fully automatic feature matching and structure from motion pipeline will sometimes fail or produce errors for some images in a collection. Currently, our system is unable to process such datasets. However in future we will provide a way to interactively correct such failures using a few user-provided correspondences (as in the classical approach of Debevec et. al. [1996]) within our system. Our system could also be extended to deal with curved and free-form surfaces in the scene and handle a wider range of architectural styles.

References

Agarwala, A., Dontcheva, M., Agrawala, M., Drucker, S., Colburn, A., Curless, B., Salesin, D., and Cohen, M. 2004. Interactive digital photomontage. In *ACM Trans. on Graphics (SIGGRAPH'04)*, 294–302.

Aliaga, D. G., Rosen, P. A., and Bekins, D. R. 2007. Style grammars for interactive visualization of architecture. *IEEE Trans. on Visualization and Computer Graphics 13*, 4, 786–797.

Baillard, C., and Zisserman, A. 2001. Automatic reconstruction of piecewise planar models from multiple views. In *CVPR*, 559–565.

Brown, M., and Lowe, D. G. 2005. Unsupervised 3d object recognition and reconstruction in unordered datasets. In *3DIM '05*, IEEE Computer Society, Washington, DC, USA, 56–63.

Cipolla, R., and Robertson, D. 1999. 3d models of architectural scenes from uncalibrated images and vanishing points. *ICIAP 00*.

Criminisi, A., Reid, I. D., and Zisserman, A. 2000. Single view metrology. *Int. J. of Computer Vision 40*, 2, 123–148.

Debevec, P. E., Taylor, C. J., and Malik, J. 1996. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In *SIGGRAPH 1996, Computer Graphics Proceedings*, 11–20.

Dick, A. R., Torr, P. H. S., and Cipolla, R. 2004. Modelling and interpretation of architecture from several images. *Int. J. Comput. Vision 60*, 2, 111–134.

El-Hakim, S., Whiting, E., and Gonzo, L. 2005. 3d modeling with reusable and integrated building blocks. *The 7th Conference on Optical 3-D Measurement Techniques*.

Fischler, M. A., and Bolles, R. C. 1981. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM 24*, 6, 381–395.

Gibson, S., Hubbold, R., Cook, J., and Howard, T. 2003. Interactive reconstruction of virtual environments from video sequences. *Computers and Graphics 27*, 2.

Goesele, M., Snavely, N., Curless, B., Hoppe, H., and Seitz, S. M. 2007. Multi-view stereo for community photo collections. In *ICCV*.

Hoiem, D., Efros, A. A., and Hebert, M. 2005. Automatic photo pop-up. In *SIGGRAPH 2005, Computer Graphics Proceedings*.

Igarashi, T., and Hughes, J. F. 2001. A suggestive interface for 3d drawing. In *UIST*, 173–181.

Kim, S. J., and Pollefeys, M. 2008. Robust radiometric calibration and vignetting correction. *IEEE Transactions on Pattern Analysis and Machine Intelligence 30*, 4, 562–576.

Lempitsky, V. S., and Ivanov, D. V. 2007. Seamless mosaicing of image-based texture maps. In *CVPR*.

Mueller, P., Zeng, G., Wonka, P., and Gool, L. V. 2007. Image-based procedural modeling of facades. *ACM Trans. on Graphics (SIGGRAPH 2007) 26*, 3, 85:1–85:9.

Oh, B. M., Chen, M., Dorsey, J., and Durand, F. 2001. Image-based modeling and photo editing. In *SIGGRAPH 2001, Computer Graphics Proceedings*, 433–442.

PÉREZ, P., GANGNET, M., AND BLAKE, A. 2003. Poisson image editing. *ACM Trans. on Graphics (SIGGRAPH'03) 22*, 3, 313–318.

POLLEFEYS, M., GOOL, L. V., VERGAUWEN, M., VERBIEST, F., CORNELIS, K., TOPS, J., AND KOCH, R. 2004. Visual modeling with a hand-held camera. *Int. J. of Comput. Vision 59*, 3, 207–232.

POLLEFEYS, M., NISTER, D., FRAHM, J.-M., AKBARZADEH, A., MORDOHAI, P., ET AL. 2008. Detailed real-time urban 3d reconstruction from video. *Int. J. of Computer Vision* (in press).

ROTHER, C. 2002. A new approach for vanishing point detection in architectural environments. In *BMVC*, vol. 20, 382–391.

SKETCHUP. http://www.sketchup.com.

SNAVELY, N., SEITZ, S. M., AND SZELISKI, R. 2006. Photo tourism: exploring photo collections in 3d. *ACM Trans. on Graphics (SIGGRAPH'06)*, 835–846.

TRIGGS, B., MCLAUCHLAN, P., HARTLEY, R., AND FITZGIBBON, A. 2000. Bundle adjustment – A modern synthesis. In *Vision Algorithms: Theory and Practice*, W. Triggs, A. Zisserman, and R. Szeliski, Eds., LNCS. Springer Verlag, 298–375.

VAN DEN HENGEL, A., DICK, A., THORMAHLEN, T., TORR, P. H. S., AND B.WARD. 2006. Fitting multiple models to multiple images with minimal user interaction. *In the Intl. Workshop on the Representation and use of Prior Knowledge in Vision*.

VAN DEN HENGEL, A., DICK, A., THORMÄHLEN, T., WARD, B., AND TORR, P. H. S. 2007. Videotrace: rapid interactive scene modelling from video. *ACM Trans. on Graphics (SIGGRAPH'07)*, 86.

WERNER, T., AND ZISSERMAN, A. 2002. New techniques for automated architecture reconstruction from photographs. In *ECCV (2)*, 541–555.

WILCZKOWIAK, M., STURM, P., AND BOYER, E. 2005. Using geometric constraints through parallelepipeds for calibration and 3d modeling. *IEEE Trans. on PAMI 27*, 2, 194–207.

ZELEZNIK, R., HERNDON, K., AND HUGHES, J. 1996. Sketch: An interface for sketching 3D scenes. *ACM Trans. on Graphics (SIGGRAPH'96)*, 163–170.

ZIEGLER, R., MATUSIK, W., PFISTER, H., AND MCMILLAN, L. 2003. 3d reconstruction using labeled image regions. In *SGP '03: Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, 248–259.

A Estimating Vanishing Points

Our single image VP estimation method uses RANSAC. Two random samples are selected from the set of detected line segments. Their intersection produces a VP hypothesis. To evaluate this hypothesis, we test its agreement with all the line segments using the method of Rother [2002]. Once a good RANSAC solution is obtained, we repeat the process on the remaining outliers and look for up to N unique solutions (N = 8 in our experiments). This approach is quite fast and reliably finds the VPs with good support (high inlier count of concurrent line segments). However some spurious solutions are also obtained, especially when a set of lines is accidentally concurrent but does not actually correspond to parallel lines in 3D.

Therefore we globally optimize the camera parameters and estimate three orthogonal directions in a subsequent bundle adjustment. This minimizes the alignment error between VPs and their associated family of parallel lines along with the reprojection error of the reconstructed 3D points. The camera parameters obtained earlier are used to backproject all the single-view VP estimates into directions which are then clustered on a unit sphere. The up-vector is computed by detecting the cluster that is most well aligned with the up vector for most of the cameras. Once clusters corresponding to three orthogonal directions have been detected, the bundle adjustment is initialized with the cluster centers and the existing camera parameters. This bundle minimizes the sum of the reprojection error of 3D points reconstructed by structure from motion and the alignment error of the line segments w.r.t. the corresponding vanishing points (the criteria described by Rother [2002]). Along with refining the camera parameters, three orthogonal directions in the scene are computed. When projected into the images, these produce globally consistent orthogonal vanishing points in all the images.

B Constrained Vertex Positions

For editing the geometric model under a set of constraints, we use Lagrange Multipliers to update the position of the vertices of our model which are constrained to lie on 1–3 planes. In the following section, we denote the plane equation by $\vec{n}.x$ + d = 0 where $\vec{n}$ represents the normal vector of the plane and d represents its distance from the origin.

In order to constrain a vertex $\vec{p} = (p_x, p_y, p_z)$ to lie on two planes $\vec{n}_1 = (n_{1x}, n_{1y}, n_{1z}), d_1$ and $\vec{n}_2 = (n_{2x}, n_{2y}, n_{2z}), d_2$, the 3D point p must satisfy

$$\begin{aligned} \vec{n}_1.p + d_1 &= 0 \\ \vec{n}_2.p + d_2 &= 0 \end{aligned}$$

We also minimize the distance between $\vec{p}$ and a reference point $\vec{p}_o = (p_{ox}, p_{oy}, p_{oz})$, which is given by the following expression.

$$|\vec{p} - \vec{p}_o|^2 = (p_x - p_{ox})^2 + (p_y - p_{oy})^2 + (p_z - p_{oz})^2$$

Minimizing this objective function under the two constraints with Lagrange Multipliers λ and μ, we obtain a linear system of equations. This linear system is solved to obtain the position of the constrained vertex.

$$\begin{bmatrix} 2 & 0 & 0 & n_{1x} & n_{2x} \\ 0 & 2 & 0 & n_{1y} & n_{2y} \\ 0 & 0 & 2 & n_{1z} & n_{2z} \\ n_{1x} & n_{1y} & n_{1z} & 0 & 0 \\ n_{2x} & n_{2y} & n_{2z} & 0 & 0 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ \lambda \\ \mu \end{bmatrix} = \begin{bmatrix} 2p_{ox} \\ 2p_{oy} \\ 2p_{oz} \\ -d_1 \\ -d_2 \end{bmatrix}$$

Note that a vertex lying on three non-coplanar planes is fully constrained to lie at the point of intersection of the three planes.